```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
```

```
DDDDDDDD    BBBBBBBB      GGGGGGGG    SSSSSSSS    CCCCCCCC  RRRRRRRR    EEEEEEEEEE  EEEEEEEEEE  NN      NN
DDDDDDDD    BBBBBBBB      GGGGGGGG    SSSSSSSS    CCCCCCCC  RRRRRRRR    EEEEEEEEEE  EEEEEEEEEE  NN      NN
DD      DD  BB      BB  GG          SS          CC          RR      RR  EE          EE          NN      NN
DD      DD  BB      BB  GG          SS          CC          RR      RR  EE          EE          NN      NN
DD      DD  BB      BB  GG          SS          CC          RR      RR  EE          EE          NNNN    NN
DD      DD  BB      BB  GG          SS          CC          RR      RR  EE          EE          NNNN    NN
DD      DD  BBBBBBBB    GG            SSSSSS    CC          RRRRRRRR    EEEEEEEE    EEEEEEEE    NN  NN  NN
DD      DD  BBBBBBBB    GG            SSSSSS    CC          RRRRRRRR    EEEEEEEE    EEEEEEEE    NN  NN  NN
DD      DD  BB      BB  GG  GGGGGG          SS  CC          RR  RR      EE          EE          NN    NNNN
DD      DD  BB      BB  GG  GGGGGG          SS  CC          RR  RR      EE          EE          NN    NNNN
DD      DD  BB      BB  GG      GG          SS  CC          RR    RR    EE          EE          NN      NN      ....
DD      DD  BB      BB  GG      GG          SS  CC          RR    RR    EE          EE          NN      NN      ....
DDDDDDDD    BBBBBBBB      GGGGGG    SSSSSSSS      CCCCCCCC  RR      RR  EEEEEEEEEE  EEEEEEEEEE  NN      NN      ....
DDDDDDDD    BBBBBBBB      GGGGGG    SSSSSSSS      CCCCCCCC  RR      RR  EEEEEEEEEE  EEEEEEEEEE  NN      NN      ....

        LL          IIIIII      SSSSSSSS
        LL          IIIIII      SSSSSSSS
        LL            II      SS
        LL            II      SS
        LL            II      SS
        LL            II      SS
        LL            II        SSSSSS
        LL            II        SSSSSS
        LL            II              SS
        LL            II              SS
        LL            II              SS
        LL            II              SS
        LLLLLLLLLL  IIIIII    SSSSSSSS
        LLLLLLLLLL  IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE DBGSCREEN (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !**************************************************************************
;     6          0006  1 !*                                                                        *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                  *
;    10          0010  1 !*                                                                        *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                           *
;    17          0017  1 !*                                                                        *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                           *
;    21          0021  1 !*                                                                        *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !**************************************************************************
;    27          0027  1 !
;    28          0028  1 ! WRITTEN BY
;    29          0029  1 !       Bert Beander    March, 1983
;    30          0030  1 !
;    31          0031  1 ! MODULE FUNCTION
;    32          0032  1 !       This module contains all routines which implement DEBUG's
;    33          0033  1 !       Screen Mode facility.
;    34          0034  1 !
;    35          0035  1
;    36          0036  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    37          0170  1
;    38          0171  1 FORWARD ROUTINE
;    39          0172  1     DBG$KEY_INITIALIZE: NOVALUE,          ! Initialize keypad definitions
;    40          0173  1     DBG$SCR_CREATE_DISPLAY,               ! Create a new Screen Display Entry
;    41          0174  1     DBG$SCR_CREATE_WINDOW,                ! Create a new Screen Window Entry
;    42          0175  1     DBG$SCR_DELETE_DISPLAY,               ! Delete a specified screen display
;    43          0176  1     DBG$SCR_DELETE_WINDOW,                ! Delete a specified screen window
;    44          0177  1     DBG$SCR_DISPLAY_COMMAND,              ! Process the DISPLAY command semantics
;    45          0178  1     DBG$SCR_EMPTY_DISPLAY: NOVALUE,       ! Empty a display of it contents
;    46          0179  1     DBG$SCR_EXECUTE_CANDISP_CMD:NOVALUE,  ! Execute the CANCEL DISPLAY command
;    47          0180  1     DBG$SCR_EXECUTE_CANWIND_CMD:NOVALUE,  ! Execute the CANCEL WINDOW command
;    48          0181  1     DBG$SCR_EXECUTE_DISPLAY_CMD:          ! Execute the DISPLAY and SET DISPLAY
;    49          0182  1                         NOVALUE,          !     commands
;    50          0183  1     DBG$SCR_EXECUTE_SAVE_CMD: NOVALUE,    ! Execute the SAVE command
;    51          0184  1     DBG$SCR_EXECUTE_SCROLL_CMD: NOVALUE,  ! Execute the SCROLL command
;    52          0185  1     DBG$SCR_EXECUTE_SELECT_CMD: NOVALUE,  ! Execute the SELECT command
;    53          0186  1     DBG$SCR_EXECUTE_SETTERM_CMD:NOVALUE,  ! Execute the SET TERMINAL command
;    54          0187  1     DBG$SCR_EXECUTE_SETWIND_CMD:NOVALUE,  ! Execute the SET WINDOW command
;    55          0188  1     DBG$SCR_EXECUTE_SHODISP_CMD:NOVALUE,  ! Execute the SHOW DISPLAY command
;    56          0189  1     DBG$SCR_EXECUTE_SHOSEL_CMD: NOVALUE,  ! Execute the SHOW SELECT command
;    57          0190  1     DBG$SCR_EXECUTE_SHOWIND_CMD:NOVALUE,  ! Execute the SHOW WINDOW command
```

```
;    58          0191  1     DBG$SCR_GENERATE_SCREEN: NOVALUE,    ! Generate contents of all automatically
;    59          0192  1                                          !      updated screen displays
;    60          0193  1     DBG$SCR_INITIALIZE: NOVALUE,         ! Initialize Screen Debugging code
;    61          0194  1     DBG$SCR_LOOKUP_DISPLAY,              ! Look up a screen display by its name
;    62          0195  1     DBG$SCR_LOOKUP_WINDOW,               ! Look up a screen window by its name
;    63          0196  1     DBG$SCR_OUTPUT_LINE_MINIMAL:NOVALUE, ! Output line to screen using a minimal
;    64          0197  1                                          !      screen update algorithm
;    65          0198  1     DBG$SCR_OUTPUT_SCREEN: NOVALUE,      ! Output updated screen image
;    66          0199  1     DBG$SCR_PARSE_CANDISP_CMD: NOVALUE,  ! Parse the CANCEL DISPLAY command
;    67          0200  1     DBG$SCR_PARSE_CANWIND_CMD: NOVALUE,  ! Parse the CANCEL WINDOW command
;    68          0201  1     DBG$SCR_PARSE_DISPLAY_CMD: NOVALUE,  ! Parse DISPLAY and SET DISPLAY commands
;    69          0202  1     DBG$SCR_PARSE_SAVE_CMD: NOVALUE,     ! Parse the SAVE command
;    70          0203  1     DBG$SCR_PARSE_SCROLL_CMD: NOVALUE,   ! Parse the SCROLL command
;    71          0204  1     DBG$SCR_PARSE_SELECT_CMD: NOVALUE,   ! Parse the SELECT command
;    72          0205  1     DBG$SCR_PARSE_SETTERM_CMD: NOVALUE,  ! Parse the SET TERMINAL command
;    73          0206  1     DBG$SCR_PARSE_SETWIND_CMD: NOVALUE,  ! Parse the SET WINDOW command
;    74          0207  1     DBG$SCR_PARSE_SHODISP_CMD: NOVALUE,  ! Parse the SHOW DISPLAY command
;    75          0208  1     DBG$SCR_PARSE_SHOWIND_CMD: NOVALUE,  ! Parse the SHOW WINDOW command
;    76          0209  1     DBG$SCR_READ_LINE,                   ! Read a line of input
;    77          0210  1     DBG$SCR_SCREEN_MODE: NOVALUE,        ! Execute SET MODE SCREEN and NOSCREEN
;    78          0211  1     DBG$SCR_SCREEN_NORMAL: NOVALUE,      ! Make screen normal for SPAWN command
;    79          0212  1     DBG$SCR_SCREEN_TERM: NOVALUE,        ! Make screen normal for exit handler
;    80          0213  1     DBG$SCR_SCREEN_TO_LOGFILE: NOVALUE,  ! Write screen image to DEBUG log file
;    81          0214  1     DBG$SCR_SCROLL_DISPLAY: NOVALUE,     ! Scroll a specified display
;    82          0215  1     DBG$SCR_SCROLL_SOURCE_DOWN,          ! Scroll a source display down
;    83          0216  1     DBG$SCR_SCROLL_SOURCE_UP,            ! Scroll a source display up
;    84          0217  1     DBG$SCR_SOURCE_BEGIN: NOVALUE,       ! Begin retrieval of source lines
;    85          0218  1     DBG$SCR_SOURCE_END: NOVALUE,         ! End retrieval of source lines
;    86          0219  1     DBG$SCR_SOURCE_LINE: NOVALUE,        ! Insert a source line in a display
;    87          0220  1     DBG$SCR_UPDATE_PASTEBOARD: NOVALUE,  ! Update contents of pasteboard
;    88          0221  1     DBG$SCR_WRITE_ERROR,                 ! Write an error message to a display
;    89          0222  1     DBG$SCR_WRITE_LINE: NOVALUE,         ! Write a line of output to a display
;    90          0223  1     FORMAT_SOURCE_LINE: NOVALUE,         ! Format a source line for display
;    91          0224  1     HANDLER_EXECUTE_SAVE,                ! Handler for EXECUTE_SAVE_CMD routine
;    92          0225  1     HANDLER_SCROLL_SOURCE,               ! Handler for SCROLL_SOURCE_xx routines
;    93          0226  1     PARSE_DISPLAY_NAME,                  ! Parse and look up a display name
;    94          0227  1     PARSE_WINDOW_NAME,                   ! Parse and look up a window name
;    95          0228  1     PARSE_WINDOW_PARAMETERS: NOVALUE,    ! Parse window specification parameters
;    96          0229  1     REDIRECT_SCREEN_OUTPUT: NOVALUE,     ! Redirect screen output to DBG$OUTPUT
;    97          0230  1     REGISTER_DISPLAY: NOVALUE,           ! Fill in contents of a register display
;    98          0231  1     REGISTER_FILL: NOVALUE,              ! Fill hex value into register display
;    99          0232  1     WINDOW_SOURCE_LINE;                  ! Find first source line number in a
;   100          0233  1                                          !      display's screen window
```

```
;  102     0234  1 EXTERNAL ROUTINE
;  103     0235  1     DBG$FINAL_HANDL,                ! DEBUG's final condition handler
;  104     0236  1     DBG$GET_MEMORY,                 ! Get a permanent memory block
;  105     0237  1     DBG$GET_TEMPMEM,                ! Get a temporary memory block
;  106     0238  1     DBG$NCIS_ADD: NOVALUE,          ! Add buffer to the Command Input Stream
;  107     0239  1     DBG$NEWLINE: NOVALUE,           ! Flush the current print line
;  108     0240  1     DBG$NMATCH,                     ! Match a specified string in parsing
;  109     0241  1     DBG$NSAVE_BREAK_BUFFER,         ! Pick up a DEBUG command list
;  110     0242  1     DBG$NSAVE_DECIMAL_INTEGER,      ! Parse a decimal integer constant
;  111     0243  1     DBG$PRINT: NOVALUE,             ! Format and output print output
;  112     0244  1     DBG$REL_MEMORY: NOVALUE,        ! Release a permanent memory block
;  113     0245  1     DBG$SRC_LNUM_RANGE: NOVALUE,    ! Get a module's line number range
;  114     0246  1     DBG$SRC_TYPE_LNUM_SOURCE: NOVALUE,  ! Type a range of source lines
;  115     0247  1     DBG$STA_SYMNAME,                ! Returns ASCIC symbol name given SYMID
;  116     0248  1     DBG$SYNTAX_ERROR: NOVALUE,      ! Report a syntax error in parsing
;  117     0249  1     DBG$WRITE_LOG_FILE: NOVALUE,    ! Write a line to the DEBUG log file
;  118     0250  1     OTS$CVT_L_TZ,                   ! Convert a longword to hexadecimal text
;  119     0251  1     SCR$DOWN_SCROLL,                ! Scroll screen down one line
;  120     0252  1     SCR$ERASE_LINE,                 ! Erase rest of line on screen
;  121     0253  1     SCR$PUT_BUFFER,                 ! Output screen buffer to terminal
;  122     0254  1     SCR$PUT_SCREEN,                 ! Output a line of text to the screen
;  123     0255  1     SCR$SCREEN_INFO,                ! Get information about screen terminal
;  124     0256  1     SCR$SET_BUFFER,                 ! Set up screen buffer mode
;  125     0257  1     SCR$SET_CURSOR,                 ! Set the cursor on the screen
;  126     0258  1     SCR$SET_OUTPUT,                 ! Redirect output to DBG$OUTPUT
;  127     0259  1     SCR$SET_SCROLL,                 ! Set up scrolling region on the screen
;  128     0260  1     SCR$UP_SCROLL,                  ! Scroll screen up one line
;  129     0261  1     SMG$ADD_KEY_DEF,                ! Add a keypad key definition
;  130     0262  1     SMG$SET_KEYPAD_MODE;            ! Set terminal to numeric or applic.
;  131     0263  1
;  132     0264  1 EXTERNAL
;  133     0265  1     DBG$GB_KEYPAD_INPUT: BYTE,      ! Keypad input flag
;  134     0266  1     DBG$GL_KEYBOARD_ID,             ! Used by SMG routines
;  135     0267  1     DBG$GB_LANGUAGE: BYTE,          ! The currently SET language code
;  136     0268  1     DBG$GL_CISHEAD: REF CIS$LINK,   ! Pointer to first CIS entry on DEBUG's
;  137     0269  1                                     !     Command Input Stream
;  138     0270  1     DBG$GL_DEVELOPER: BITVECTOR[],  ! Developer switches
;  139     0271  1     DBG$GL_KEY_TABLE_ID,            ! Keypad input Key-Table Id number
;  140     0272  1     DBG$RUNFRAME: BLOCK[,BYTE],     ! The current user run frame
;  141     0273  1     DBG$SRC_NEXT_LNUM,              ! The next source line number
;  142     0274  1     DBG$SRC_TERM_WIDTH;             ! Terminal width of output device
;  143     0275  1
;  144     0276  1 LITERAL
;  145     0277  1     CAR_RET             = 13,       ! Carriage Return character
;  146     0278  1     TAB_CHAR            = 9;        ! Horizontal Tab character
;  147     0279  1
;  148     0280  1 BIND
;  149     0281  1     DBG$CS_ALL          = UPLIT BYTE(%ASCIC 'ALL'),     ! Counted ASCII strings
;  150     0282  1     DBG$CS_AS           = UPLIT BYTE(%ASCIC 'AS'),      !      used in parsing
;  151     0283  1     DBG$CS_AT           = UPLIT BYTE(%ASCIC 'AT'),
;  152     0284  1     DBG$CS_BOTTOM       = UPLIT BYTE(%ASCIC 'BOTTOM'),
;  153     0285  1     DBG$CS_CLEAR        = UPLIT BYTE(%ASCIC 'CLEAR'),
;  154     0286  1     DBG$CS_COLON        = UPLIT BYTE(%ASCIC ':'),
;  155     0287  1     DBG$CS_COMMA        = UPLIT BYTE(%ASCIC ','),
;  156     0288  1     DBG$CS_CR           = UPLIT BYTE(1, CAR_RET),       ! Carriage return
;  157     0289  1     DBG$CS_DO           = UPLIT BYTE(%ASCIC 'DO'),
;  158     0290  1     DBG$CS_DOWN         = UPLIT BYTE(%ASCIC 'DOWN'),
```

```
;  159     0291  1      DBG$CS_EQUAL          = UPLIT BYTE(%ASCIC '='),
;  160     0292  1      DBG$CS_GENERATE       = UPLIT BYTE(%ASCIC 'GENERATE'),
;  161     0293  1      DBG$CS_HIDE           = UPLIT BYTE(%ASCIC 'HIDE'),
;  162     0294  1      DBG$CS_HISTORY        = UPLIT BYTE(%ASCIC 'HISTORY'),
;  163     0295  1      DBG$CS_INPUT          = UPLIT BYTE(%ASCIC 'INPUT'),
;  164     0296  1      DBG$CS_LEFT           = UPLIT BYTE(%ASCIC 'LEFT'),
;  165     0297  1      DBG$CS_LPAREN         = UPLIT BYTE(%ASCIC '('),
;  166     0298  1      DBG$CS_MARK_CHANGE    = UPLIT BYTE(%ASCIC 'MARK_CHANGE'),
;  167     0299  1      DBG$CS_NOMARK_CHANGE= UPLIT BYTE(%ASCIC 'NOMARK_CHANGE'),
;  168     0300  1      DBG$CS_NORMAL         = UPLIT BYTE(%ASCIC 'NORMAL'),
;  169     0301  1      DBG$CS_OUTPUT         = UPLIT BYTE(%ASCIC 'OUTPUT'),
;  170     0302  1      DBG$CS_REFRESH        = UPLIT BYTE(%ASCIC 'REFRESH'),
;  171     0303  1      DBG$CS_REGISTER       = UPLIT BYTE(%ASCIC 'REGISTER'),
;  172     0304  1      DBG$CS_REMOVED        = UPLIT BYTE(%ASCIC 'REMOVED'),
;  173     0305  1      DBG$CS_RIGHT          = UPLIT BYTE(%ASCIC 'RIGHT'),
;  174     0306  1      DBG$CS_RPAREN         = UPLIT BYTE(%ASCIC ')'),
;  175     0307  1      DBG$CS_SCROLLING      = UPLIT BYTE(%ASCIC 'SCROLLING'),
;  176     0308  1      DBG$CS_SIZE           = UPLIT BYTE(%ASCIC 'SIZE'),
;  177     0309  1      DBG$CS_SLASH          = UPLIT BYTE(%ASCIC '/'),
;  178     0310  1      DBG$CS_SOURCE         = UPLIT BYTE(%ASCIC 'SOURCE'),
;  179     0311  1      DBG$CS_TOP            = UPLIT BYTE(%ASCIC 'TOP'),
;  180     0312  1      DBG$CS_UP             = UPLIT BYTE(%ASCIC 'UP'),
;  181     0313  1      DBG$CS_WIDTH          = UPLIT BYTE(%ASCIC 'WIDTH');
;  182     0314  1
;  183     0315  1  GLOBAL
;  184     0316  1      DBG$GL_SCREEN_ERROR: INITIAL(0),     ! Pointer to active error Display Entry
;  185     0317  1      DBG$GL_SCREEN_HISTORY: INITIAL(0),   ! Pointer to active history Display Entry
;  186     0318  1      DBG$GL_SCREEN_INPUT: INITIAL(0),     ! Pointer to active input Display Entry
;  187     0319  1      DBG$GL_SCREEN_LOG: INITIAL(FALSE),   ! Flag set if screen logging is active
;  188     0320  1      DBG$GL_SCREEN_MODE: INITIAL(FALSE),  ! Flag set if screen mode is active
;  189     0321  1      DBG$GL_SCREEN_NOGO: INITIAL(FALSE),  ! Flag to turn off STEP and GO commands
;  190     0322  1      DBG$GL_SCREEN_OUTPUT: INITIAL(0),    ! Pointer to active output Display Entry
;  191     0323  1      DBG$GL_SCREEN_SOURCE: INITIAL(0);    ! Pointer to active source Display Entry
;  192     0324  1
;  193     0325  1  OWN
;  194     0326  1      BOT_OFFSET,                          ! Offset from DROW to bottom ref point
;  195     0327  1      BOT_REF_FILEID,                      ! FileId of bottom reference point
;  196     0328  1      BOT_REF_RECNUM,                      ! Record number of bottom reference point
;  197     0329  1      DBG$GL_VT100_FLAG: INITIAL(TRUE),    ! Flag set if terminal is VT100 class
;  198     0330  1      DBG$SCR_CURDISP_INPUT:               ! Pointer to the Display Entry for the
;  199     0331  1          REF DBG$DISP_ENTRY INITIAL(0),   !     current input display
;  200     0332  1      DBG$SCR_CURDISP_OUTPUT:              ! Pointer to the Display Entry for the
;  201     0333  1          REF DBG$DISP_ENTRY INITIAL(0),   !     current output display
;  202     0334  1      DBG$SCR_CURDISP_SCROLL:              ! Pointer to the Display Entry for the
;  203     0335  1          REF DBG$DISP_ENTRY INITIAL(0),   !     current scrolling display
;  204     0336  1      DBG$SCR_CURDISP_SOURCE:              ! Pointer to the Display Entry for the
;  205     0337  1          REF DBG$DISP_ENTRY INITIAL(0),   !     current source line display
;  206     0338  1      DBG$SCR_DISPLAY_LIST:                ! List head for doubly linked list of
;  207     0339  1          VECTOR[2,LONG] INITIAL(0,0),     !     Screen Display Entries
;  208     0340  1      DBG$SCR_WINDOW_LIST:                 ! List head for doubly linked list of
;  209     0341  1          VECTOR[2,LONG] INITIAL(0,0),     !     Screen Window Entries
;  210     0342  1      FIXED_SCROLL_AMOUNT,                 ! Flag set if fixed scrolling amount is
;  211     0343  1                                           !     used for source scrolling
;  212     0344  1      FIXED_SCROLL_VALID: INITIAL(FALSE),  ! Flag set if above flag is valid
;  213     0345  1      INVSCR_FLAG,                         ! Flag to invalidate scrolling count
;  214     0346  1                                           !     for scrolling displays
;  215     0347  1      MARKLINE_DISABLE_FLAG:               ! Flag set to disable marking of source
```

```
;  216     0348  1                                INITIAL(FALSE),     !      line with "->" when scrolling
;  217     0349  1      OLD_CNT: VECTOR[DBG$K_PASTE_SIZE],          ! Old screen image char count per line
;  218     0350  1      OLD_REND:                                   ! Old screen image rendition bits
;  219     0351  1          VECTOR[DBG$K_PASTE_SIZE, BYTE],         !      per line
;  220     0352  1      OLD_SCREEN: VECTOR[                         ! Old screen image used to implement a
;  221     0353  1          DBG$K_PASTE_SIZE*132, BYTE],            !      minimal screen update scheme
;  222     0354  1      OLD_SCREEN_REND: VECTOR[                    ! Old screen image rendition codes used
;  223     0355  1          DBG$K_PASTE_SIZE*132, BYTE],            !      for minimal screen update scheme
;  224     0356  1      OLD_VALID:                                  ! Flags to indicate old screen image
;  225     0357  1          BITVECTOR[DBG$K_PASTE_SIZE],            !      text for a line is valid
;  226     0358  1      PASTEBOARD: DBG$PASTEBOARD,                 ! The terminal screen pasteboard
;  227     0359  1      SAVED_SCROLL,                               ! Saved scrolling count when scrolling
;  228     0360  1                                                  !      a source line display
;  229     0361  1      SCREEN_MODE_FIRST_TIME:                     ! Flag cleared the first time screen
;  230     0362  1                                INITIAL(TRUE),    !      mode is set
;  231     0363  1      SCROLL_AMOUNT,                              ! Source display scrolling amount from
;  232     0364  1                                                  !      SCROLL command
;  233     0365  1      SCROLL_TO_BOTTOM,                           ! Flag set when scrolling to bottom of
;  234     0366  1                                                  !      a source display
;  235     0367  1      TOP_OFFSET,                                 ! Offset from DROW to top ref point
;  236     0368  1      TOP_REF_FILEID,                             ! FileId of top reference point in display
;  237     0369  1      TOP_REF_RECNUM;                             ! Record number of top reference point
```

```
;  239          0370  1 ! MACRO TO CREATE A KEYPAD DEFINITION
;  240          0371  1 !
;  241          0372  1 !
;  242          0373  1 !
;  243          0374  1 ! This macro creates a built-in keypad definition.  It is called as follows:
;  244          0375  1 !
;  245          0376  1 !       KEYPAD_DEF(IF_STATE, KEYNAME, FLAGS, COMMAND, SET_STATE)
;  246          0377  1 !
;  247          0378  1 ! where IF_STATE is a string giving the "color key" that must have been pressed
;  248          0379  1 ! first, KEYNAME is a string giving the name of the key being defined, FLAGS is
;  249          0380  1 ! a bit-mask that indicates whether the key is echoed and whether it terminates
;  250          0381  1 ! the current input operation, COMMAND is the DEBUG command string to which the
;  251          0382  1 ! new key maps, and SET_STATE is a string giving the new "color state" that is
;  252          0383  1 ! set by the new key (if any).  IF_STATE and SET_STATE can both be null param-
;  253          0384  1 ! eters, and COMMAND can be a null string.
;  254          0385  1 !
;  255          0386  1 MACRO
;  256        M 0387  1     KEYPAD_DEF(IF_STATE, KEYNAME, FLAGS, COMMAND, SET_STATE) =
;  257        M 0388  1         BEGIN
;  258        M 0389  1
;  259        M 0390  1
;  260        M 0391  1         !Set up the pointers to the string descriptors for If-state and the
;  261        M 0392  1         ! Set-state parameters.  For the default "color" state, these pointers
;  262        M 0393  1         ! are zero; otherwise they point to descriptors for 'GOLD' or 'BLUE'.
;  263        M 0394  1         !
;  264        M 0395  1         %IF %NULL(IF_STATE)           %THEN IF_PTR = 0
;  265        M 0396  1         %ELSE %IF IF_STATE EQL 'GOLD' %THEN IF_PTR = GOLD_DESC
;  266        M 0397  1         %ELSE %IF IF_STATE EQL 'BLUE' %THEN IF_PTR = BLUE_DESC
;  267        M 0398  1         %FI %FI %FI;
;  268        M 0399  1         %IF %NULL(SET_STATE)           %THEN SET_PTR = 0
;  269        M 0400  1         %ELSE %IF SET_STATE EQL 'GOLD' %THEN SET_PTR = GOLD_DESC
;  270        M 0401  1         %ELSE %IF SET_STATE EQL 'BLUE' %THEN SET_PTR = BLUE_DESC
;  271        M 0402  1         %FI %FI %FI;
;  272        M 0403  1
;  273        M 0404  1
;  274        M 0405  1         ! Set up the string descriptors for the key-name and the command
;  275        M 0406  1         ! string parameters.  Also set up the attributes longword with the
;  276        M 0407  1         ! noecho and noterminal flags.
;  277        M 0408  1         !
;  278        M 0409  1         KEY_DESC[DSC$W_LENGTH] = %CHARCOUNT(KEYNAME);
;  279        M 0410  1         KEY_DESC[DSC$A_POINTER] = UPLIT BYTE(%ASCII KEYNAME);
;  280        M 0411  1         CMD_DESC[DSC$W_LENGTH] = %CHARCOUNT(COMMAND);
;  281        M 0412  1         CMD_DESC[DSC$A_POINTER] = UPLIT BYTE(%ASCII COMMAND);
;  282        M 0413  1         ATTRIBUTES = FLAGS;
;  283        M 0414  1
;  284        M 0415  1
;  285        M 0416  1         ! Call the Screen Management routine that defines a new keypad key.
;  286        M 0417  1         !
;  287        M 0418  1         STATUS = SMG$ADD_KEY_DEF(DBG$GL_KEY_TABLE_ID, KEY_DESC, .IF_PTR,
;  288        M 0419  1                                  ATTRIBUTES, CMD_DESC, .SET_PTR);
;  289        M 0420  1         IF NOT .STATUS THEN SIGNAL(.STATUS);
;  290          0421  1         END %;
```

```
;  292     0422  1 GLOBAL ROUTINE DBG$KEY_INITIALIZE: NOVALUE =
;  293     0423  1 !
;  294     0424  1 ! FUNCTION
;  295     0425  1 !       This routine initializes the keypad by creating all of the "built-in"
;  296     0426  1 !       predeclared keypad definitions.  Each key definition is done by a call
;  297     0427  1 !       on the KEYPAD_DEF macro which then sets up the appropriate call on the
;  298     0428  1 !       SMG$ADD_KEY_DEF screen management routine to actually add the keypad
;  299     0429  1 !       definition.
;  300     0430  1 !
;  301     0431  1 ! INPUTS
;  302     0432  1 !       NONE
;  303     0433  1 !
;  304     0434  1 ! OUTPUTS
;  305     0435  1 !       NONE
;  306     0436  1 !
;  307     0437  1
;  308     0438  2     BEGIN
;  309     0439  2
;  310     0440  2     LITERAL
;  311     0441  2         DEFAULT         = 0,            ! The default keypad state (no color)
;  312     0442  2         GOLD            = 1,            ! The GOLD keypad state
;  313     0443  2         BLUE            = 2;            ! The BLUE keypad state
;  314     0444  2
;  315     0445  2     LITERAL
;  316     0446  2         ECHO_TERM       = 2,            ! Echo command and terminate input
;  317     0447  2         NOECHO_TERM     = 3,            ! Do not echo, but terminate input
;  318     0448  2         ECHO_NOTERM     = 0;            ! Echo, but do not terminate input
;  319     0449  2
;  320     0450  2     LOCAL
;  321     0451  2         ATTRIBUTES,                     ! Longword containing attribute flags
;  322     0452  2         BLUE_DESC: BLOCK[8,BYTE],       ! String descriptor for BLUE keyword
;  323     0453  2         CMD_DESC: BLOCK[8,BYTE],        ! DEBUG command string descriptor
;  324     0454  2         GOLD_DESC: BLOCK[8,BYTE],       ! String descriptor for GOLD keyword
;  325     0455  2         IF_PTR,                         ! Pointer to If-state string descriptor
;  326     0456  2         KEY_DESC: BLOCK[8,BYTE],        ! Keyname string descriptor
;  327     0457  2         SET_PTR,                        ! Pointer to Set-state string descriptor
;  328     0458  2         STATUS;                         ! Status code from the SMG routine
;  329     0459  2
;  330     0460  2
;  331     0461  2     ! Initialize all the string descriptors.
;  332     0462  2     !
;  333     0463  2     GOLD_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  334     0464  2     GOLD_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  335     0465  2     GOLD_DESC[DSC$W_LENGTH] = 4;
;  336     0466  2     GOLD_DESC[DSC$A_POINTER] = UPLIT BYTE(%ASCII 'GOLD');
;  337     0467  2     BLUE_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  338     0468  2     BLUE_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  339     0469  2     BLUE_DESC[DSC$W_LENGTH] = 4;
;  340     0470  2     BLUE_DESC[DSC$A_POINTER] = UPLIT BYTE(%ASCII 'BLUE');
;  341     0471  2     KEY_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  342     0472  2     KEY_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  343     0473  2     CMD_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  344     0474  2     CMD_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  345     0475  2
;  346     0476  2
;  347     0477  2     ! Define all "built-in" keypad keys which are pre-defined by DEBUG.
;  348     0478  2     ! First define the color keys GOLD and BLUE.
```

```
;  349          0479  2      !
;  350          0480  2      KEYPAD_DEF(,          'PF1', ECHO_NOTERM,    '', 'GOLD');
;  351          0481  2      KEYPAD_DEF('GOLD',    'PF1', ECHO_NOTERM,    '', 'GOLD');
;  352          0482  2      KEYPAD_DEF('BLUE',    'PF1', ECHO_NOTERM,    '', 'GOLD');
;  353          0483  2      KEYPAD_DEF(,          'PF4', ECHO_NOTERM,    '', 'BLUE');
;  354          0484  2      KEYPAD_DEF('BLUE',    'PF4', ECHO_NOTERM,    '', 'BLUE');
;  355          0485  2      KEYPAD_DEF('GOLD',    'PF4', ECHO_NOTERM,    '', 'BLUE');
;  356          0486  2
;  357          0487  2
;  358          0486  2      ! Define the PF2 and PF3 keys in the first row of keypad keys.
;  359          0489  2      !
;  360          0490  2      KEYPAD_DEF(,          'PF2', ECHO_TERM,      'Help Keypad Nocolor');
;  361          0491  2      KEYPAD_DEF('GOLD',    'PF2', ECHO_TERM,      'Help Keypad Gold');
;  362          0492  2      KEYPAD_DEF('BLUE',    'PF2', ECHO_TERM,      'Help Keypad Blue');
;  363          0493  2      KEYPAD_DEF(,          'PF3', ECHO_TERM,      'Set Mode Screen');
;  364          0494  2      KEYPAD_DEF('GOLD',    'PF3', ECHO_TERM,      'Set Mode Noscreen');
;  365          0495  2      KEYPAD_DEF('BLUE',    'PF3', NOECHO_TERM,    'Display/Generate');
;  366          0496  2
;  367          0497  2
;  368          0498  2      ! Define the second row of keypad keys (keys 7, 8, 9, and MINUS).
;  369          0499  2      !
;  370          0500  2      KEYPAD_DEF(,          'KP8', NOECHO_TERM,    'Scroll/Up');
;  371          0501  2      KEYPAD_DEF('GOLD',    'KP8', NOECHO_TERM,    'Scroll/Top');
;  372          0502  2      KEYPAD_DEF('BLUE',    'KP8', ECHO_NOTERM,    'Scroll/Up ');
;  373          0503  2      KEYPAD_DEF(,          'KP9', NOECHO_TERM,    'Display %Nextdisp');
;  374        P 0504  2      KEYPAD_DEF(,          'MINUS', NOECHO_TERM,
;  375          0505  2                            'Display %Nextdisp At Fs; Select/Scroll %Curdisp');
;  376        P 0506  2      KEYPAD_DEF('BLUE',    'MINUS', NOECHO_TERM,
;  377          0507  2                            'Display Src At H1, Out At H2; Select/Scroll Src');
;  378          0508  2
;  379          0509  2
;  380          0510  2      ! Display the third row of keypad keys (keys 4, 5, 6, and COMMA).
;  381          0511  2      !
;  382          0512  2      KEYPAD_DEF(,          'KP4', NOECHO_TERM,    'Scroll/Left');
;  383          0513  2      KEYPAD_DEF('GOLD',    'KP4', NOECHO_TERM,    'Scroll/Left:132');
;  384          0514  2      KEYPAD_DEF('BLUE',    'KP4', ECHO_NOTERM,    'Scroll/Left ');
;  385          0515  2      KEYPAD_DEF(,          'KP5', NOECHO_TERM,    'Examine/Source .0\%PC');
;  386          0516  2      KEYPAD_DEF('GOLD',    'KP5', ECHO_TERM,      'Show Calls');
;  387          0517  2      KEYPAD_DEF('BLUE',    'KP5', ECHO_TERM,      'Show Calls 3');
;  388          0518  2      KEYPAD_DEF(,          'KP6', NOECHO_TERM,    'Scroll/Right');
;  389          0519  2      KEYPAD_DEF('BLUE',    'KP6', ECHO_NOTERM,    'Scroll/Right ');
;  390          0520  2      KEYPAD_DEF(,          'COMMA', ECHO_TERM,    'Go');
;  391          0521  2
;  392          0522  2
;  393          0523  2      ! Define the fourth row of keypad keys (keys 1, 2, and 3).
;  394          0524  2      !
;  395          0525  2      KEYPAD_DEF(,          'KP1', ECHO_TERM,      'Examine');
;  396          0526  2      KEYPAD_DEF(,          'KP2', NOECHO_TERM,    'Scroll/Down');
;  397          0527  2      KEYPAD_DEF('GOLD',    'KP2', NOECHO_TERM,    'Scroll/Bottom');
;  398          0528  2      KEYPAD_DEF('BLUE',    'KP2', ECHO_NOTERM,    'Scroll/Down ');
;  399          0529  2      KEYPAD_DEF(,          'KP3', NOECHO_TERM,    'Select/Scroll %Nextscroll');
;  400          0530  2      KEYPAD_DEF('GOLD',    'KP3', NOECHO_TERM,    'Select/Output %Nextoutput');
;  401          0531  2      KEYPAD_DEF('BLUE',    'KP3', NOECHO_TERM,    'Select/Source %Nextsource');
;  402          0532  2
;  403          0533  2
;  404          0534  2      ! Define the fifth row of keypad keys (keys 0 and PERIOD).
;  405          0535  2      !
```

```
;  406     0536  2     KEYPAD_DEF(,        'KP0', ECHO_TERM,     'Step');
;  407     0537  2     KEYPAD_DEF('GOLD', 'KP0', ECHO_TERM,     'Step/Into');
;  408     0538  2     KEYPAD_DEF('BLUE', 'KP0', ECHO_TERM,     'Step/Over');
;  409     0539  2     KEYPAD_DEF(,        'PERIOD', ECHO_NOTERM, '');
;  410     0540  2     KEYPAD_DEF('GOLD', 'PERIOD', ECHO_NOTERM, '');
;  411     0541  2     KEYPAD_DEF('BLUE', 'PERIOD', ECHO_NOTERM, '');
;  412     0542  2
;  413     0543  2
;  414     0544  2     ! Define the meanings of the control characters we use.
;  415     0545  2     !
;  416     0546  2     KEYPAD_DEF(,        'CTRLW', NOECHO_TERM, 'Display/Refresh');
;  417     0547  2     KEYPAD_DEF('GOLD', 'CTRLW', NOECHO_TERM, 'Display/Refresh');
;  418     0548  2     KEYPAD_DEF('BLUE', 'CTRLW', NOECHO_TERM, 'Display/Refresh');
;  419     0549  2
;  420     0550  2
;  421     0551  2     ! Define some of the special keys found on the LK201 keyboard.  These
;  422     0552  2     ! keys are not available on the VT100.
;  423     0553  2     !
;  424     0554  2     KEYPAD_DEF(,        'HELP', ECHO_TERM, 'Help Keypad Summary');
;  425     0555  2     KEYPAD_DEF(,        'PREV_SCREEN', NOECHO_TERM, 'Scroll/Up');
;  426     0556  2     KEYPAD_DEF(,        'NEXT_SCREEN', NOECHO_TERM, 'Scroll/Down');
;  427     0557  2     KEYPAD_DEF(,        'SELECT', NOECHO_TERM, 'Select/Scroll %Nextscroll');
;  428     0558  2
;  429     0559  2
;  430     0560  2     ! All built-in keypad keys have been defined.  Now return.
;  431     0561  2     !
;  432     0562  2     RETURN;
;  433     0563  2
;  434     0564  1     END;
```

```
                                                        .TITLE  DBGSCREEN
                                                        .IDENT  \V04-000\

                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                                00000 P.AAA:    .ASCII  <3>\ALL\                ;
                                                00004 P.AAB:    .ASCII  <2>\AS\                 ;
                                                00007 P.AAC:    .ASCII  <2>\AT\                 ;
                                                0000A P.AAD:    .ASCII  <6>\BOTTOM\             ;
                                                00011 P.AAE:    .ASCII  <5>\CLEAR\              ;
                                                00017 P.AAF:    .ASCII  <1>\:\                  ;
                                                00019 P.AAG:    .ASCII  <1>\,\                  ;
                                                0001B P.AAH:    .BYTE   1, 13                   ;
                                                0001D P.AAI:    .ASCII  <2>\DO\                 ;
                                                00020 P.AAJ:    .ASCII  <4>\DOWN\               ;
                                                00025 P.AAK:    .ASCII  <1>\=\                  ;
                                                00027 P.AAL:    .ASCII  <8>\GENERATE\           ;
                                                00030 P.AAM:    .ASCII  <4>\HIDE\               ;
                                                00035 P.AAN:    .ASCII  <7>\HISTORY\            ;
                                                0003D P.AAO:    .ASCII  <5>\INPUT\              ;
                                                00043 P.AAP:    .ASCII  <4>\LEFT\               ;
                                                00048 P.AAQ:    .ASCII  <1>\(\                  ;
                                                0004A P.AAR:    .ASCII  <11>\MARK_CHANGE\       ;
                                                00056 P.AAS:    .ASCII  <13>\NOMARK_CHANGE\     ;
                                                00064 P.AAT:    .ASCII  <6>\NORMAL\             ;
                                                0006B P.AAU:    .ASCII  <6>\OUTPUT\             ;
```

```
                                48 53 45 52 46 45 52 07  00072 P.AAV:   .ASCII  <7>\REFRESH\
                             52 45 54 53 49 47 45 52 08  0007A P.AAW:   .ASCII  <8>\REGISTER\
                                44 45 56 4F 4D 45 52 07  00083 P.AAX:   .ASCII  <7>\REMOVED\
                                   54 48 47 49 52 05  0008B P.AAY:   .ASCII  <5>\RIGHT\
                                               29 01  00091 P.AAZ:   .ASCII  <1>\)\
                          47 4E 49 4C 4C 4F 52 43 53 09  00093 P.ABA:   .ASCII  <9>\SCROLLING\
                                      45 5A 49 53 04  0009D P.ABB:   .ASCII  <4>\SIZE\
                                               2F 01  000A2 P.ABC:   .ASCII  <1>\/\
                                45 43 52 55 4F 53 06  000A4 P.ABD:   .ASCII  <6>\SOURCE\
                                         50 4F 54 03  000AB P.ABE:   .ASCII  <3>\TOP\
                                            50 55 02  000AF P.ABF:   .ASCII  <2>\UP\
                                   48 54 44 49 57 05  000B2 P.ABG:   .ASCII  <5>\WIDTH\
                                         44 4C 4F 47  000B8 P.ABH:   .ASCII  \GOLD\
                                         45 55 4C 42  000BC P.ABI:   .ASCII  \BLUE\
                                            31 46 50  000C0 P.ABJ:   .ASCII  \PF1\
                                                      000C3 P.ABK:   .BLKB   0
                                            31 46 50  000C3 P.ABL:   .ASCII  \PF1\
                                                      000C6 P.ABM:   .BLKB   0
                                            31 46 50  000C6 P.ABN:   .ASCII  \PF1\
                                                      000C9 P.ABO:   .BLKB   0
                                            34 46 50  000C9 P.ABP:   .ASCII  \PF4\
                                                      000CC P.ABQ:   .BLKB   C
                                            34 46 50  000CC P.ABR:   .ASCII  \PF4\
                                                      000CF P.ABS:   .BLKB   0
                                            34 46 50  000CF P.ABT:   .ASCII  \PF4\
                                                      000D2 P.ABU:   .BLKB   0
                                            32 46 50  000D2 P.ABV:   .ASCII  \PF2\
63 6F 4E 20 64 61 70 79 65 4B 20 70 6C 65 48  000D5 P.ABW:   .ASCII  \Help Keypad Nocolor\
                                         72 6F 6C 6F  000E4
                                            32 46 50  000E8 P.ABX:   .ASCII  \PF2\
6C 6F 47 20 64 61 70 79 65 4B 20 70 6C 65 48  000EB P.ABY:   .ASCII  \Help Keypad Gold\
                                                  64  000FA
                                            32 46 50  000FB P.ABZ:   .ASCII  \PF2\
75 6C 42 20 64 61 70 79 65 4B 20 70 6C 65 48  000FE P.ACA:   .ASCII  \Help Keypad Blue\
                                                  65  0010D
                                            33 46 50  0010E P.ACB:   .ASCII  \PF3\
6E 65 65 72 63 53 20 65 64 6F 4D 20 74 65 53  00111 P.ACC:   .ASCII  \Set Mode Screen\
                                            33 46 50  00120 P.ACD:   .ASCII  \PF3\
65 72 63 73 6F 4E 20 65 64 6F 4D 20 74 65 53  00123 P.ACE:   .ASCII  \Set Mode Noscreen\
                                               6E 65  00132
                                            33 46 50  00134 P.ACF:   .ASCII  \PF3\
74 61 72 65 6E 65 47 2F 79 61 6C 70 73 69 44  00137 P.ACG:   .ASCII  \Display/Generate\
                                                  65  00146
                                            38 50 4B  00147 P.ACH:   .ASCII  \KP8\
                                70 55 2F 6C 6C 6F 72 63 53  0014A P.ACI:   .ASCII  \Scroll/Up\
                                            38 50 4B  00153 P.ACJ:   .ASCII  \KP8\
                             70 6F 54 2F 6C 6C 6F 72 63 53  00156 P.ACK:   .ASCII  \Scroll/Top\
                                            38 50 4B  00160 P.ACL:   .ASCII  \KP8\
                             20 70 55 2F 6C 6C 6F 72 63 53  00163 P.ACM:   .ASCII  \Scroll/Up \
                                            39 50 4B  0016D P.ACN:   .ASCII  \KP9\
69 64 74 78 65 4E 25 20 79 61 6C 70 73 69 44  00170 P.ACO:   .ASCII  \Display %Nextdisp\
                                               70 73  0017F
                                         53 55 4E 49 4D  00181 P.ACP:   .ASCII  \MINUS\
69 64 74 78 65 4E 25 20 79 61 6C 70 73 69 44  00186 P.ACQ:   .ASCII  \Display %Nextdisp At Fs; Select/Scroll %\
63 65 6C 65 53 20 38 73 46 20 74 41 20 70 73  00195
                          25 20 6C 6C 6F 72 63 53 2F 74  001A4
                                   70 73 69 64 72 75 43  001AE          .ASCII  \Curdisp\
```

```
                                                53 55 4E 49 4D 001B5 P.ACR:  .ASCII  \MINUS\
20 74 41 20 63 72 53 20 79 61 6C 70 73 69 44 001BA P.ACS:  .ASCII  \Display Src At H1, Out At H2; Select/Scr\
20 3B 32 48 20 74 41 20 74 75 4F 20 2C 31 48 001C9
                   72 63 53 2F 74 63 65 6C 65 53 001D8
                                  63 72 53 20 6C 6C 6F 001E2          .ASCII  \oll Src\
                                              34 50 4B 001E9 P.ACT:  .ASCII  \KP4\
                      74 66 65 4C 2F 6C 6C 6F 72 63 53 001EC P.ACU:  .ASCII  \Scroll/Left\
                                              34 50 4B 001F7 P.ACV:  .ASCII  \KP4\
32 33 31 3A 74 66 65 4C 2F 6C 6C 6F 72 63 53 001FA P.ACW:  .ASCII  \Scroll/Left:132\
                                              34 50 4B 00209 P.ACX:  .ASCII  \KP4\
                   20 74 66 65 4C 2F 6C 6C 6F 72 63 53 0020C P.ACY:  .ASCII  \Scroll/Left \
                                              35 50 4B 00218 P.ACZ:  .ASCII  \KP5\
20 65 63 72 75 6F 53 2F 65 6E 69 6D 61 78 45 0021B P.ADA:  .ASCII  \Examine/Source .0\<92>\%PC\
                                  43 50 25 5C 30 2E 0022A
                                              35 50 4B 00230 P.ADB:  .ASCII  \KP5\
                         73 6C 6C 61 43 20 77 6F 68 53 00233 P.ADC:  .ASCII  \Show Calls\
                                              35 50 4B 0023D P.ADD:  .ASCII  \KP5\
                   33 20 73 6C 6C 61 43 20 77 6F 68 53 00240 P.ADE:  .ASCII  \Show Calls 3\
                                              36 50 4B 0024C P.ADF:  .ASCII  \KP6\
                   74 68 67 69 52 2F 6C 6C 6F 72 63 53 0024F P.ADG:  .ASCII  \Scroll/Right\
                                              36 50 4B 0025B P.ADH:  .ASCII  \KP6\
                20 74 68 67 69 52 2F 6C 6C 6F 72 63 53 0025E P.ADI:  .ASCII  \Scroll/Right \
                                        41 4D 4D 4F 43 0026B P.ADJ:  .ASCII  \COMMA\
                                              6F 47 00270 P.ADK:  .ASCII  \Go\
                                              31 50 4B 00272 P.ADL:  .ASCII  \KP1\
                                  65 6E 69 6D 61 78 45 00275 P.ADM:  .ASCII  \Examine\
                                              32 50 4B 0027C P.ADN:  .ASCII  \KP2\
                      6E 77 6F 44 2F 6C 6C 6F 72 63 53 0027F P.ADO:  .ASCII  \Scroll/Down\
                                              32 50 4B 0028A P.ADP:  .ASCII  \KP2\
                6D 6F 74 74 6F 42 2F 6C 6C 6F 72 63 53 0028D P.ADQ:  .ASCII  \Scroll/Bottom\
                                              32 50 4B 0029A P.ADR:  .ASCII  \KP2\
                   20 6E 77 6F 44 2F 6C 6C 6F 72 63 53 0029D P.ADS:  .ASCII  \Scroll/Down \
                                              33 50 4B 002A9 P.ADT:  .ASCII  \KP3\
25 20 6C 6C 6F 72 63 53 2F 74 63 65 6C 65 53 002AC P.ADU:  .ASCII  \Select/Scroll %Nextscroll\
                         6C 6C 6F 72 63 73 74 78 65 4E 002BB
                                              33 50 4B 002C5 P.ADV:  .ASCII  \KP3\
25 20 74 75 70 74 75 4F 2F 74 63 65 6C 65 53 002C8 P.ADW:  .ASCII  \Select/Output %Nextoutput\
                         74 75 70 74 75 6F 74 78 65 4E 002D7
                                              33 50 4B 002E1 P.ADX:  .ASCII  \KP3\
25 20 65 63 72 75 6F 53 2F 74 63 65 6C 65 53 002E4 P.ADY:  .ASCII  \Select/Source %Nextsource\
                         65 63 72 75 6F 73 74 78 65 4E 002F3
                                              30 50 4B 002FD P.ADZ:  .ASCII  \KP0\
                                        70 65 74 53 00300 P.AEA:  .ASCII  \Step\
                                              30 50 4B 00304 P.AEB:  .ASCII  \KP0\
                            6F 74 6E 49 2F 70 65 74 53 00307 P.AEC:  .ASCII  \Step/Into\
                                              30 50 4B 00310 P.AED:  .ASCII  \KP0\
                            72 65 76 4F 2F 70 65 74 53 00313 P.AEE:  .ASCII  \Step/Over\
                                        44 4F 49 52 45 50 0031C P.AEF:  .ASCII  \PERIOD\
                                              00322 P.AEG:  .BLKB   0
                                        44 4F 49 52 45 50 00322 P.AEH:  .ASCII  \PERIOD\
                                              00328 P.AEI:  .BLKB   0
                                        44 4F 49 52 45 50 00328 P.AEJ:  .ASCII  \PERIOD\
                                              0032E P.AEK:  .BLKB   0
                                        57 4C 52 54 43 0032E P.AEL:  .ASCII  \CTRLW\
68 73 65 72 66 65 52 2F 79 61 6C 70 73 69 44 00333 P.AEM:  .ASCII  \Display/Refresh\
                                        57 4C 52 54 43 00342 P.AEN:  .ASCII  \CTRLW\
68 73 65 72 66 65 52 2F 79 61 6C 70 73 69 44 00347 P.AEO:  .ASCII  \Display/Refresh\
```

```
                                                
```

```
                        .BLKB   4
00000001  01758 SCREEN_MODE_FIRST_TIME:
                        .LONG   1                                       ;
          0175C SCROLL_AMOUNT:
                        .BLKB   4
          01760 SCROLL_TO_BOTTOM:
                        .BLKB   4
          01764 TOP_OFFSET:
                        .BLKB   4
          01768 TOP_REF_FILEID:
                        .BLKB   4
          0176C TOP_REF_RECNUM:
                        .BLKB   4

                        .PSECT  DBG$GLOBAL,NOEXE,  PIC,2

00000000  00000 DBG$GL_SCREEN_ERROR::
                        .LONG   0                                       ;
00000000  00004 DBG$GL_SCREEN_HISTORY::
                        .LONG   0                                       ;
00000000  00008 DBG$GL_SCREEN_INPUT::
                        .LONG   0                                       ;
00000000  0000C DBG$GL_SCREEN_LOG::
                        .LONG   0                                       ;
00000000  00010 DBG$GL_SCREEN_MODE::
                        .LONG   0                                       ;
00000000  00014 DBG$GL_SCREEN_NOGO::
                        .LONG   0                                       ;
00000000  00018 DBG$GL_SCREEN_OUTPUT::
                        .LONG   0                                       ;
00000000  0001C DBG$GL_SCREEN_SOURCE::
                        .LONG   0                                       ;
                        .LONG   0                                       ;

DBG$CS_ALL=             P.AAA
DBG$CS_AS=              P.AAB
DBG$CS_AT=              P.AAC
DBG$CS_BOTTOM=          P.AAD
DBG$CS_CLEAR=           P.AAE
DBG$CS_COLON=           P.AAF
DBG$CS_COMMA=           P.AAG
DBG$CS_CR=              P.AAH
DBG$CS_DO=              P.AAI
DBG$CS_DOWN=            P.AAJ
DBG$CS_EQUAL=           P.AAK
DBG$CS_GENERATE=        P.AAL
DBG$CS_HIDE=            P.AAM
DBG$CS_HISTORY=         P.AAN
DBG$CS_INPUT=           P.AAO
DBG$CS_LEFT=            P.AAP
DBG$CS_LPAREN=          P.AAQ
DBG$CS_MARK_CHANGE=     P.AAR
DBG$CS_NOMARK_CHANGE=
                        P.AAS
DBG$CS_NORMAL=          P.AAT
DBG$CS_OUTPUT=          P.AAU
DBG$CS_REFRESH=         P.AAV
DBG$CS_REGISTER=        P.AAW
```

```
                DBG$CS_REMOVED=     P.AAX
                DBG$CS_RIGHT=       P.AAY
                DBG$CS_RPAREN=      P.AAZ
                DBG$CS_SCROLLING=   P.ABA
                DBG$CS_SIZE=        P.ABB
                DBG$CS_SLASH=       P.ABC
                DBG$CS_SOURCE=      P.ABD
                DBG$CS_TOP=         P.ABE
                DBG$CS_UP=          P.ABF
                DBG$CS_WIDTH=       P.ABG
                        .EXTRN  DBG$FINAL_HANDL
                        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
                        .EXTRN  DBG$NCIS_ADD, DBG$NEWLINE
                        .EXTRN  DBG$NMATCH, DBG$NSAVE_BREAK_BUFFER
                        .EXTRN  DBG$NSAVE_DECIMAL_INTEGER
                        .EXTRN  DBG$PRINT, DBG$REL_MEMORY
                        .EXTRN  DBG$SRC_LNUM_RANGE
                        .EXTRN  DBG$SRC_TYPE_LNUM_SOURCE
                        .EXTRN  DBG$STA_SYMNAME
                        .EXTRN  DBG$SYNTAX_ERROR
                        .EXTRN  DBG$WRITE_LOG_FILE
                        .EXTRN  OTS$CVT_L_TZ, SCR$DOWN_SCROLL
                        .EXTRN  SCR$ERASE_LINE, SCR$PUT_BUFFER
                        .EXTRN  SCR$PUT_SCREEN, SCR$SCREEN_INFO
                        .EXTRN  SCR$SET_BUFFER, SCR$SET_CURSOR
                        .EXTRN  SCR$SET_OUTPUT, SCR$SET_SCROLL
                        .EXTRN  SCR$UP_SCROLL, SMG$ADD_KEY_DEF
                        .EXTRN  SMG$SET_KEYPAD_MODE
                        .EXTRN  DBG$GB_KEYPAD_INPUT
                        .EXTRN  DBG$GL_KEYBOARD_ID
                        .EXTRN  DBG$GB_LANGUAGE
                        .EXTRN  DBG$GL_CISHEAD, DBG$GL_DEVELOPER
                        .EXTRN  DBG$GL_KEY_TABLE_ID
                        .EXTRN  DBG$RUNFRAME, DBG$SRC_NEXT_LNUM
                        .EXTRN  DBG$SRC_TERM_WIDTH

                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                01FC 00000      .ENTRY  DBG$KEY_INITIALIZE, Save R2,R3,R4,R5,R6,R7,- ; 0422
                                                        R8                                          ;
     58 00000000G   00  9E 00002        MOVAB   LIB$SIGNAL, R8                                      ;
     57 00000000G   00  9E 00009        MOVAB   SMG$ADD_KEY_DEF, R7                                 ;
     56 00000000G   00  9E 00010        MOVAB   DBG$GL_KEY_TABLE_ID, R0                             ;
     55 00000000'   EF  9E 00017        MOVAB   P.ABK, R5                                           ;
     5E             1C  C2 0001E        SUBL2   #28, SP                                             ;
04   AE 010E0004   8F  D0 00021        MOVL    #17694724, GOLD_DESC                                ; 0465
08   AE       F5   A5  9E 00029        MOVAB   P.ABH, GOLD_DESC+4                                  ; 0466
14   AE 010E0004   8F  D0 0002E        MOVL    #17694724, BLUE_DESC                                ; 0469
18   AE       F9   A5  9E 00036        MOVAB   P.ABI, BLUE_DESC+4                                  ; 0470
0E   AE     010E   8F  B0 0003B        MOVW    #270, CMD_DESC+2                                    ; 0474
                    54  D4 00041        CLRL    IF_PTR                                              ; 0480
     53       04   AE  9E 00043        MOVAB   GOLD_DESC, SET_PTR                                  ;
         010E0003   8F  DD 00047        PUSHL   #17694723                                           ;
04   AE       FD   A5  9E 0004D        MOVAB   P.ABJ, KEY_DESC+4                                   ;
                10   AE  B4 00052        CLRW    CMD_DESC                                            ;
14   AE           65  9E 00055        MOVAB   P.ABK, CMD_DESC+4                                   ;
                  7E  D4 00059        CLRL    ATTRIBUTES                                          ;
```

```
                        53  DD 0005B          PUSHL   SET_PTR                                    ;
                    18  AE  9F 0005D          PUSHAB  CMD_DESC                                   ;
                    08  AE  9F 00060          PUSHAB  ATTRIBUTES                                 ;
                        54  DD 00063          PUSHL   IF_PTR                                     ;
                    14  AE  9F 00065          PUSHAB  KEY_DESC                                   ;
                        56  DD 00068          PUSHL   R6                                         ;
                67              06  FB 0006A          CALLS   #6, SMG$ADD_KEY_DEF                ;
                52              50  D0 0006D          MOVL    R0, STATUS                         ;
                05              52  E8 00070          BLBS    STATUS, 1$                         ;
                                52  DD 00073          PUSHL   STATUS                             ;
                68              01  FB 00075          CALLS   #1, LIB$SIGNAL                     ;
                54      0C  AE  9E 00078  1$:      MOVAB   GOLD_DESC, IF_PTR                  ;  0481
                53      0C  AE  9E 0007C          MOVAB   GOLD_DESC, SET_PTR                 ;
            04  AE              03  B0 00080          MOVW    #3, KEY_DESC                       ;
            08  AE              65  9E 00084          MOVAB   P.ABL, KEY_DESC+4                  ;
                    14  AE      B4 00088          CLRW    CMD_DESC                           ;
            18  AE      03      A5  9E 0008B          MOVAB   P.ABM, CMD_DESC+4                  ;
                                6E  D4 00090          CLRL    ATTRIBUTES                         ;
                                53  DD 00092          PUSHL   SET_PTR                            ;
                    18  AE      9F 00094          PUSHAB  CMD_DESC                           ;
                    08  AE      9F 00097          PUSHAB  ATTRIBUTES                         ;
                                54  DD 0009A          PUSHL   IF_PTR                             ;
                    14  AE      9F 0009C          PUSHAB  KEY_DESC                           ;
                                56  DD 0009F          PUSHL   R6                                 ;
                67              06  FB 000A1          CALLS   #6, SMG$ADD_KEY_DEF                ;
                52              50  D0 000A4          MOVL    R0, STATUS                         ;
                05              52  E8 000A7          BLBS    STATUS, 2$                         ;
                                52  DD 000AA          PUSHL   STATUS                             ;
                68              01  FB 000AC          CALLS   #1, LIB$SIGNAL                     ;
                54      1C  AE  9E 000AF  2$:      MOVAB   BLUE_DESC, IF_PTR                  ;  0482
                53      0C  AE  9E 000B3          MOVAB   GOLD_DESC, SET_PTR                 ;
            04  AE              03  B0 000B7          MOVW    #3, KEY_DESC                       ;
            08  AE      03      A5  9E 000BB          MOVAB   P.ABN, KEY_DESC+4                  ;
                    14  AE      B4 000C0          CLRW    CMD_DESC                           ;
            18  AE      06      A5  9E 000C3          MOVAB   P.ABO, CMD_DESC+4                  ;
                                6E  D4 000C8          CLRL    ATTRIBUTES                         ;
                                53  DD 000CA          PUSHL   SET_PTR                            ;
                    18  AE      9F 000CC          PUSHAB  CMD_DESC                           ;
                    08  AE      9F 000CF          PUSHAB  ATTRIBUTES                         ;
                                54  DD 000D2          PUSHL   IF_PTR                             ;
                    14  AE      9F 000D4          PUSHAB  KEY_DESC                           ;
                                56  DD 000D7          PUSHL   R6                                 ;
                67              06  FB 000D9          CALLS   #6, SMG$ADD_KEY_DEF                ;
                52              50  D0 000DC          MOVL    R0, STATUS                         ;
                05              52  E8 000DF          BLBS    STATUS, 3$                         ;
                                52  DD 000E2          PUSHL   STATUS                             ;
                68              01  FB 000E4          CALLS   #1, LIB$SIGNAL                     ;
                                54  D4 000E7  3$:      CLRL    IF_PTR                             ;  0483
                53      1C  AE  9E 000E9          MOVAB   BLUE_DESC, SET_PTR                 ;
            04  AE              03  B0 000ED          MOVW    #3, KEY_DESC                       ;
            08  AE      06      A5  9E 000F1          MOVAB   P.ABP, KEY_DESC+4                  ;
                    14  AE      B4 000F6          CLRW    CMD_DESC                           ;
            18  AE      09      A5  9E 000F9          MOVAB   P.ABQ, CMD_DESC+4                  ;
                                6E  D4 000FE          CLRL    ATTRIBUTES                         ;
                                53  DD 00100          PUSHL   SET_PTR                            ;
                    18  AE      9F 00102          PUSHAB  CMD_DESC                           ;
                    08  AE      9F 00105          PUSHAB  ATTRIBUTES                         ;
```

```
                        54   DD 00108          PUSHL   IF_PTR
                    14  AE   9F 0010A          PUSHAB  KEY_DESC
                        56   DD 0010D          PUSHL   R6
                    67       06   FB 0010F     CALLS   #6, SMG$ADD_KEY_DEF
                    52       50   D0 00112     MOVL    R0, STATUS
                    05       52   E8 00115     BLBS    STATUS, 4$
                             52   DD 00118     PUSHL   STATUS
                    68       01   FB 0011A     CALLS   #1, LIB$SIGNAL
                    54  1C   AE   9E 0011D 4$:  MOVAB   BLUE_DESC, IF_PTR                    0484
                    53  1C   AE   9E 00121     MOVAB   BLUE_DESC, SET_PTR
                04  AE       03   B0 00125     MOVW    #3, KEY_DESC
                08  AE   09  A5   9E 00129     MOVAB   P.ABR, KEY_DESC+4
                        14   AE   B4 0012E     CLRW    CMD_DESC
                18  AE   0C  A5   9E 00131     MOVAB   P.ABS, CMD_DESC+4
                        6E   D4   00136        CLRL    ATTRIBUTES
                        53   DD 00138          PUSHL   SET_PTR
                    18  AE   9F 0013A          PUSHAB  CMD_DESC
                    08  AE   9F 0013D          PUSHAB  ATTRIBUTES
                        54   DD 00140          PUSHL   IF_PTR
                    14  AE   9F 00142          PUSHAB  KEY_DESC
                        56   DD 00145          PUSHL   R6
                    67       06   FB 00147     CALLS   #6, SMG$ADD_KEY_DEF
                    52       50   D0 0014A     MOVL    R0, STATUS
                    05       52   E8 0014D     BLBS    STATUS, 5$
                             52   DD 00150     PUSHL   STATUS
                    68       01   FB 00152     CALLS   #1, LIB$SIGNAL
                    54  0C   AE   9E 00155 5$:  MOVAB   GOLD_DESC, IF_PTR                    0485
                    53  1C   AE   9E 00159     MOVAB   BLUE_DESC, SET_PTR
                04  AE       03   B0 0015D     MOVW    #3, KEY_DESC
                08  AE   0C  A5   9E 00161     MOVAB   P.ABT, KEY_DESC+4
                        14   AE   B4 00166     CLRW    CMD_DESC
                18  AE   0F  A5   9E 00169     MOVAB   P.ABU, CMD_DESC+4
                        6E   D4   0016E        CLRL    ATTRIBUTES
                        53   DD 00170          PUSHL   SET_PTR
                    18  AE   9F 00172          PUSHAB  CMD_DESC
                    08  AE   9F 00175          PUSHAB  ATTRIBUTES
                        54   DD 00178          PUSHL   IF_PTR
                    14  AE   9F 0017A          PUSHAB  KEY_DESC
                        56   DD 0017D          PUSHL   R6
                    67       06   FB 0017F     CALLS   #6, SMG$ADD_KEY_DEF
                    52       50   D0 00182     MOVL    R0, STATUS
                    05       52   E8 00185     BLBS    STATUS, 6$
                             52   DD 00188     PUSHL   STATUS
                    68       01   FB 0018A     CALLS   #1, LIB$SIGNAL
                             53   7C 0018D 6$:  CLRQ    SET_PTR                            0490
                04  AE       03   B0 0018F     MOVW    #3, KEY_DESC
                08  AE   0F  A5   9E 00193     MOVAB   P.ABV, KEY_DESC+4
                14  AE       13   B0 00198     MOVW    #19, CMD_DESC
                18  AE   12  A5   9E 0019C     MOVAB   P.ABW, CMD_DESC+4
                    6E       02   D0 001A1     MOVL    #2, ATTRIBUTES
                             53   DD 001A4     PUSHL   SET_PTR
                    18  AE   9F 001A6          PUSHAB  CMD_DESC
                    08  AE   9F 001A9          PUSHAB  ATTRIBUTES
                        54   DD 001AC          PUSHL   IF_PTR
                    14  AE   9F 001AE          PUSHAB  KEY_DESC
                        56   DD 001B1          PUSHL   R6
                    67       06   FB 001B3     CALLS   #6, SMG$ADD_KEY_DEF
```

```
                52              50  D0 001B6          MOVL    R0, STATUS                              ;
                05              52  E8 001B9          BLBS    STATUS, 7$                              ;
                                52  DD 001BC          PUSHL   STATUS                                  ;
                68              01  FB 001BE          CALLS   #1, LIB$SIGNAL                          ;
                54        0C    AE  9E 001C1 7$:     MOVAB   GOLD_DESC, IF_PTR                       ; 0491
                                53  D4 001C5          CLRL    SET_PTR                                 ;
          04    AE              03  B0 001C7          MOVW    #3, KEY_DESC                            ;
          08    AE        25    A5  9E 001CB          MOVAB   P.ABX, KEY_DESC+4                       ;
          14    AE              10  B0 001D0          MOVW    #16, CMD_DESC                           ;
          18    AE        28    A5  9E 001D4          MOVAB   P.ABY, CMD_DESC+4                       ;
                6E              02  D0 001D9          MOVL    #2, ATTRIBUTES                          ;
                                53  DD 001DC          PUSHL   SET_PTR                                 ;
                          18    AE  9F 001DE          PUSHAB  CMD_DESC                                ;
                          08    AE  9F 001E1          PUSHAB  ATTRIBUTES                              ;
                                54  DD 001E4          PUSHL   IF_PTR                                  ;
                          14    AE  9F 001E6          PUSHAB  KEY_DESC                                ;
                                56  DD 001E9          PUSHL   R6                                      ;
                67                  06  FB 001EB          CALLS   #6, SMG$ADD_KEY_DEF                 ;
                52                  50  D0 001EE          MOVL    R0, STATUS                          ;
                05                  52  E8 001F1          BLBS    STATUS, 8$                          ;
                                    52  DD 001F4          PUSHL   STATUS                              ;
                68                  01  FB 001F6          CALLS   #1, LIB$SIGNAL                      ;
                54              1C  AE  9E 001F9 8$:     MOVAB   BLUE_DESC, IF_PTR                   ; 0492
                                    53  D4 001FD          CLRL    SET_PTR                             ;
          04    AE                  03  B0 001FF          MOVW    #3, KEY_DESC                        ;
          08    AE              38  A5  9E 00203          MOVAB   P.ABZ, KEY_DESC+4                   ;
          14    AE                  10  B0 00208          MOVW    #16, CMD_DESC                       ;
          18    AE              3B  A5  9E 0020C          MOVAB   P.ACA, CMD_DESC+4                   ;
                6E                  02  D0 00211          MOVL    #2, ATTRIBUTES                      ;
                                    53  DD 00214          PUSHL   SET_PTR                             ;
                              18    AE  9F 00216          PUSHAB  CMD_DESC                            ;
                              08    AE  9F 00219          PUSHAB  ATTRIBUTES                          ;
                                    54  DD 0021C          PUSHL   IF_PTR                              ;
                              14    AE  9F 0021E          PUSHAB  KEY_DESC                            ;
                                    56  DD 00221          PUSHL   R6                                  ;
                67                  06  FB 00223          CALLS   #6, SMG$ADD_KEY_DEF                 ;
                52                  50  D0 00226          MOVL    R0, STATUS                          ;
                05                  52  E8 00229          BLBS    STATUS, 9$                          ;
                                    52  DD 0022C          PUSHL   STATUS                              ;
                68                  01  FB 0022E          CALLS   #1, LIB$SIGNAL                      ;
                                    53  7C 00231 9$:     CLRQ    SET_PTR                             ; 0493
          04    AE                  03  B0 00233          MOVW    #3, KEY_DESC                        ;
          08    AE              4B  A5  9E 00237          MOVAB   P.ACB, KEY_DESC+4                   ;
          14    AE                  0F  B0 0023C          MOVW    #15, CMD_DESC                       ;
          18    AE              4E  A5  9E 00240          MOVAB   P.ACC, CMD_DESC+4                   ;
                6E                  02  D0 00245          MOVL    #2, ATTRIBUTES                      ;
                                    53  DD 00248          PUSHL   SET_PTR                             ;
                              18    AE  9F 0024A          PUSHAB  CMD_DESC                            ;
                              08    AE  9F 0024D          PUSHAB  ATTRIBUTES                          ;
                                    54  DD 00250          PUSHL   IF_PTR                              ;
                              14    AE  9F 00252          PUSHAB  KEY_DESC                            ;
                                    56  DD 00255          PUSHL   R6                                  ;
                67                  06  FB 00257          CALLS   #6, SMG$ADD_KEY_DEF                 ;
                52                  50  D0 0025A          MOVL    R0, STATUS                          ;
                05                  52  E8 0025D          BLBS    STATUS, 10$                         ;
                                    52  DD 00260          PUSHL   STATUS                              ;
                68                  01  FB 00262          CALLS   #1, LIB$SIGNAL                      ;
```

```
                54        0C    AE   9E 00265 10$:    MOVAB   GOLD_DESC, IF_PTR                       ; 0494
                                  53   D4 00269         CLRL    SET_PTR                                 ;
          04    AE                03   B0 0026B         MOVW    #3, KEY_DESC                            ;
          08    AE        5D      A5   9E 0026F         MOVAB   P.ACD, KEY_DESC+4                       ;
          14    AE                11   B0 00274         MOVW    #17, CMD_DESC                           ;
          18    AE        60      A5   9E 00278         MOVAB   P.ACE, CMD_DESC+4                       ;
                6E                02   D0 0027D         MOVL    #2, ATTRIBUTES                          ;
                                  53   DD 00280         PUSHL   SET_PTR                                 ;
                            18    AE   9F 00282         PUSHAB  CMD_DESC                                ;
                            08    AE   9F 00285         PUSHAB  ATTRIBUTES                              ;
                                  54   DD 00288         PUSHL   IF_PTR                                  ;
                            14    AE   9F 0028A         PUSHAB  KEY_DESC                                ;
                                  56   DD 0028D         PUSHL   R6                                      ;
                67                      06   FB 0028F         CALLS   #6, SMG$ADD_KEY_DEF             ;
                52                      50   D0 00292         MOVL    R0, STATUS                      ;
                05                      52   E8 00295         BLBS    STATUS, 11$                     ;
                                        52   DD 00298         PUSHL   STATUS                          ;
                68                      01   FB 0029A         CALLS   #1, LIB$SIGNAL                  ;
                54        1C    AE   9E 0029D 11$:    MOVAB   BLUE_DESC, IF_PTR                       ; 0495
                                  53   D4 002A1         CLRL    SET_PTR                                 ;
          04    AE                03   B0 002A3         MOVW    #3, KEY_DESC                            ;
          08    AE        71      A5   9E 002A7         MOVAB   P.ACF, KEY_DESC+4                       ;
          14    AE                10   B0 002AC         MOVW    #16, CMD_DESC                           ;
          18    AE        74      A5   9E 002B0         MOVAB   P.ACG, CMD_DESC+4                       ;
                6E                03   D0 002B5         MOVL    #3, ATTRIBUTES                          ;
                                  53   DD 002B8         PUSHL   SET_PTR                                 ;
                            18    AE   9F 002BA         PUSHAB  CMD_DESC                                ;
                            08    AE   9F 002BD         PUSHAB  ATTRIBUTES                              ;
                                  54   DD 002C0         PUSHL   IF_PTR                                  ;
                            14    AE   9F 002C2         PUSHAB  KEY_DESC                                ;
                                  56   DD 002C5         PUSHL   R6                                      ;
                67                      06   FB 002C7         CALLS   #6, SMG$ADD_KEY_DEF             ;
                52                      50   D0 002CA         MOVL    R0, STATUS                      ;
                05                      52   E8 002CD         BLBS    STATUS, 12$                     ;
                                        52   DD 002D0         PUSHL   STATUS                          ;
                68                      01   FB 002D2         CALLS   #1, LIB$SIGNAL                  ;
                                  53   7C 002D5 12$:    CLRQ    SET_PTR                                 ; 0500
          04    AE                03   B0 002D7         MOVW    #3, KEY_DESC                            ;
          08    AE      0084      C5   9E 002DB         MOVAB   P.ACH, KEY_DESC+4                       ;
          14    AE                09   B0 002E1         MOVW    #9, CMD_DESC                            ;
          18    AE      0087      C5   9E 002E5         MOVAB   P.ACI, CMD_DESC+4                       ;
                6E                03   D0 002EB         MOVL    #3, ATTRIBUTES                          ;
                                  53   DD 002EE         PUSHL   SET_PTR                                 ;
                            18    AE   9F 002F0         PUSHAB  CMD_DESC                                ;
                            08    AE   9F 002F3         PUSHAB  ATTRIBUTES                              ;
                                  54   DD 002F6         PUSHL   IF_PTR                                  ;
                            14    AE   9F 002F8         PUSHAB  KEY_DESC                                ;
                                  56   DD 002FB         PUSHL   R6                                      ;
                67                      06   FB 002FD         CALLS   #6, SMG$ADD_KEY_DEF             ;
                52                      50   D0 00300         MOVL    R0, STATUS                      ;
                05                      52   E8 00303         BLBS    STATUS, 13$                     ;
                                        52   DD 00306         PUSHL   STATUS                          ;
                68                      01   FB 00308         CALLS   #1, LIB$SIGNAL                  ;
                54        0C    AE   9E 0030B 13$:    MOVAB   GOLD_DESC, IF_PTR                       ; 0501
                                  53   D4 0030F         CLRL    SET_PTR                                 ;
          04    AE                03   B0 00311         MOVW    #3, KEY_DESC                            ;
          08    AE      0090      C5   9E 00315         MOVAB   P.ACJ, KEY_DESC+4                       ;
```

```
                14  AE          0A  B0 0031B          MOVW    #10, CMD_DESC
                18  AE    0093  C5  9E 0031F          MOVAB   P.ACK, CMD_DESC+4
                6E            03  D0 00325          MOVL    #3, ATTRIBUTES
                              53  DD 00328          PUSHL   SET_PTR
                        18    AE  9F 0032A          PUSHAB  CMD_DESC
                        08    AE  9F 0032D          PUSHAB  ATTRIBUTES
                              54  DD 00330          PUSHL   IF_PTR
                        14    AE  9F 00332          PUSHAB  KEY_DESC
                              56  DD 00335          PUSHL   R6
                67            06  FB 00337          CALLS   #6, SMG$ADD_KEY_DEF
                52            50  D0 0033A          MOVL    R0, STATUS
                05            52  E8 0033D          BLBS    STATUS, 14$
                              52  DD 00340          PUSHL   STATUS
                68            01  FB 00342          CALLS   #1, LIB$SIGNAL
                54      1C    AE  9E 00345 14$:      MOVAB   BLUE_DESC, IF_PTR                       ; 0502
                              53  D4 00349          CLRL    SET_PTR
                04    AE            03  B0 0034B          MOVW    #3, KEY_DESC
                08    AE    009D    C5  9E 0034F          MOVAB   P.ACL, KEY_DESC+4
                14    AE            0A  B0 00355          MOVW    #10, CMD_DESC
                18    AE    00A0    C5  9E 00359          MOVAB   P.ACM, CMD_DESC+4
                6E                  D4 0035F          CLRL    ATTRIBUTES
                              53  DD 00361          PUSHL   SET_PTR
                        18    AE  9F 00363          PUSHAB  CMD_DESC
                        08    AE  9F 00366          PUSHAB  ATTRIBUTES
                              54  DD 00369          PUSHL   IF_PTR
                        14    AE  9F 0036B          PUSHAB  KEY_DESC
                              56  DD 0036E          PUSHL   R6
                67            06  FB 00370          CALLS   #6, SMG$ADD_KEY_DEF
                52            50  D0 00373          MOVL    R0, STATUS
                05            52  E8 00376          BLBS    STATUS, 15$                             ; F
                              52  DD 00379          PUSHL   STATUS
                68            01  FB 0037B          CALLS   #1, LIB$SIGNAL
                              53  7C 0037E 15$:      CLRQ    SET_PTR                                 ; 0503
                04    AE            03  B0 00380          MOVW    #3, KEY_DESC
                08    AE    00AA    C5  9E 00384          MOVAB   P.ACN, KEY_DESC+4
                14    AE            11  B0 0038A          MOVW    #17, CMD_DESC
                18    AE    00AD    C5  9E 0038E          MOVAB   P.ACO, CMD_DESC+4
                6E                  03  D0 00394          MOVL    #3, ATTRIBUTES
                              53  DD 00397          PUSHL   SET_PTR
                        18    AE  9F 00399          PUSHAB  CMD_DESC
                        08    AE  9F 0039C          PUSHAB  ATTRIBUTES
                              54  DD 0039F          PUSHL   IF_PTR
                        14    AE  9F 003A1          PUSHAB  KEY_DESC
                              56  DD 003A4          PUSHL   R6
                67            06  FB 003A6          CALLS   #6, SMG$ADD_KEY_DEF
                52            50  D0 003A9          MOVL    R0, STATUS
                05            52  E8 003AC          BLBS    STATUS, 16$
                              52  DD 003AF          PUSHL   STATUS
                68            01  FB 003B1          CALLS   #1, LIB$SIGNAL
                              53  7C 003B4 16$:      CLRQ    SET_PTR                                 ; 0505
                04    AE            05  B0 003B6          MOVW    #5, KEY_DESC
                08    AE    00BE    C5  9E 003BA          MOVAB   P.ACP, KEY_DESC+4
                14    AE            2F  B0 003C0          MOVW    #47, CMD_DESC
                18    AE    00C3    C5  9E 003C4          MOVAB   P.ACQ, CMD_DESC+4
                6E                  03  D0 003CA          MOVL    #3, ATTRIBUTES
                              53  DD 003CD          PUSHL   SET_PTR
                        18    AE  9F 003CF          PUSHAB  CMD_DESC
```

```
                08 AE 9F 003D2           PUSHAB  ATTRIBUTES
                   54 DD 003D5           PUSHL   IF_PTR
                14 AE 9F 003D7           PUSHAB  KEY_DESC
                   56 DD 003DA           PUSHL   R6
             67    06 FB 003DC           CALLS   #6, SMG$ADD_KEY_DEF
             52    50 D0 003DF           MOVL    R0, STATUS
             05    52 E8 003E2           BLBS    STATUS, 17$
                   52 DD 003E5           PUSHL   STATUS
             68    01 FB 003E7           CALLS   #1, LIB$SIGNAL
             54  1C AE 9E 003EA 17$:     MOVAB   BLUE_DESC, IF_PTR                0507
                   53 D4 003EE           CLRL    SET_PTR
       04    AE    05 B0 003F0           MOVW    #5, KEY_DESC
       08    AE 00F2 C5 9E 003F4           MOVAB   P.ACR, KEY_DESC+4
       14    AE    2F B0 003FA           MOVW    #47, CMD_DESC
       18    AE 00F7 C5 9E 003FE           MOVAB   P.ACS, CMD_DESC+4
             6E    03 D0 00404           MOVL    #3, ATTRIBUTES
                   53 DD 00407           PUSHL   SET_PTR
                18 AE 9F 00409           PUSHAB  CMD_DESC
                08 AE 9F 0040C           PUSHAB  ATTRIBUTES
                   54 DD 0040F           PUSHL   IF_PTR
                14 AE 9F 00411           PUSHAB  KEY_DESC
                   56 DD 00414           PUSHL   R6
             67    06 FB 00416           CALLS   #6, SMG$ADD_KEY_DEF
             52    50 D0 00419           MOVL    R0, STATUS
             05    52 E8 0041C           BLBS    STATUS, 18$
                   52 DD 0041F           PUSHL   STATUS
             68    01 FB 00421           CALLS   #1, LIB$SIGNAL
                   53 7C 00424 18$:      CLRQ    SET_PTR                          0512
       04    AE    03 B0 00426           MOVW    #3, KEY_DESC
       08    AE 0126 C5 9E 0042A           MOVAB   P.ACT, KEY_DESC+4
       14    AE    0B B0 00430           MOVW    #11, CMD_DESC
       18    AE 0129 C5 9E 00434           MOVAB   P.ACU, CMD_DESC+4
             6E    03 D0 0043A           MOVL    #3, ATTRIBUTES
                   53 DD 0043D           PUSHL   SET_PTR
                18 AE 9F 0043F           PUSHAB  CMD_DESC
                08 AE 9F 00442           PUSHAB  ATTRIBUTES
                   54 DD 00445           PUSHL   IF_PTR
                14 AE 9F 00447           PUSHAB  KEY_DESC
                   56 DD 0044A           PUSHL   R6
             67    06 FB 0044C           CALLS   #6, SMG$ADD_KEY_DEF
             52    50 D0 0044F           MOVL    R0, STATUS
             05    52 E8 00452           BLBS    STATUS, 19$
                   52 DD 00455           PUSHL   STATUS
             68    01 FB 00457           CALLS   #1, LIB$SIGNAL
             54  0C AE 9E 0045A 19$:     MOVAB   GOLD_DESC, IF_PTR                0513
                   53 D4 0045E           CLRL    SET_PTR
       04    AE    03 B0 00460           MOVW    #3, KEY_DESC
       08    AE 0134 C5 9E 00464           MOVAB   P.ACV, KEY_DESC+4
       14    AE    0F B0 0046A           MOVW    #15, CMD_DESC
       18    AE 0137 C5 9E 0046E           MOVAB   P.ACW, CMD_DESC+4
             6E    03 D0 00474           MOVL    #3, ATTRIBUTES
                   53 DD 00477           PUSHL   SET_PTR
                18 AE 9F 00479           PUSHAB  CMD_DESC
                08 AE 9F 0047C           PUSHAB  ATTRIBUTES
                   54 DD 0047F           PUSHL   IF_PTR
                14 AE 9F 00481           PUSHAB  KEY_DESC
                   56 DD 00484           PUSHL   R6
```

```
                67              06 FB 00486          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                52              50 D0 00489          MOVL    R0, STATUS                             ;
                05              52 E8 0048C          BLBS    STATUS, 20$                            ;
                                52 DD 0048F          PUSHL   STATUS                                 ;
                68              01 FB 00491          CALLS   #1, LIB$SIGNAL                         ;
                54        1C    AE 9E 00494 20$:     MOVAB   BLUE_DESC, IF_PTR                      ; 0514
                                53 D4 00498          CLRL    SET_PTR                                ;
          04    AE              03 B0 0049A          MOVW    #3, KEY_DESC                           ;
          08    AE        0146  C5 9E 0049E          MOVAB   P.ACX, KEY_DESC+4                      ;
          14    AE              0C B0 004A4          MOVW    #12, CMD_DESC                          ;
          18    AE        0149  C5 9E 004A8          MOVAB   P.ACY, CMD_DESC+4                      ;
                6E              D4 004AE             CLRL    ATTRIBUTES                             ;
                                53 DD 004B0          PUSHL   SET_PTR                                ;
                          18    AE 9F 004B2          PUSHAB  CMD_DESC                               ;
                          08    AE 9F 004B5          PUSHAB  ATTRIBUTES                             ;
                                54 DD 004B8          PUSHL   IF_PTR                                 ;
                          14    AE 9F 004BA          PUSHAB  KEY_DESC                               ;
                                56 DD 004BD          PUSHL   R6                                     ;
                67              06 FB 004BF          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                52              50 D0 004C2          MOVL    R0, STATUS                             ;
                05              52 E8 004C5          BLBS    STATUS, 21$                            ;
                                52 DD 004C8          PUSHL   STATUS                                 ;
                68              01 FB 004CA          CALLS   #1, LIB$SIGNAL                         ;
                                53 7C 004CD 21$:     CLRQ    SET_PTR                                ; 0515
          04    AE              03 B0 004CF          MOVW    #3, KEY_DESC                           ;
          08    AE        0155  C5 9E 004D3          MOVAB   P.ACZ, KEY_DESC+4                      ;
          14    AE              15 B0 004D9          MOVW    #21, CMD_DESC                          ;
          18    AE        0158  C5 9E 004DD          MOVAB   P.ADA, CMD_DESC+4                      ;
                6E              03 D0 004E3          MOVL    #3, ATTRIBUTES                         ;
                                53 DD 004E6          PUSHL   SET_PTR                                ;
                          18    AE 9F 004E8          PUSHAB  CMD_DESC                               ;
                          08    AE 9F 004EB          PUSHAB  ATTRIBUTES                             ;
                                54 DD 004EE          PUSHL   IF_PTR                                 ;
                          14    AE 9F 004F0          PUSHAB  KEY_DESC                               ;
                                56 DD 004F3          PUSHL   R6                                     ;
                67              06 FB 004F5          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                52              50 D0 004F8          MOVL    R0, STATUS                             ;
                05              52 E8 004FB          BLBS    STATUS, 22$                            ;
                                52 DD 004FE          PUSHL   STATUS                                 ;
                68              01 FB 00500          CALLS   #1, LIB$SIGNAL                         ;
                54        0C    AE 9E 00503 22$:     MOVAB   GOLD_DESC, IF_PTR                      ; 0516
                                53 D4 00507          CLRL    SET_PTR                                ;
          04    AE              03 B0 00509          MOVW    #3, KEY_DESC                           ;
          08    AE        016D  C5 9E 0050D          MOVAB   P.ADB, KEY_DESC+4                      ;
          14    AE              0A B0 00513          MOVW    #10, CMD_DESC                          ;
          18    AE        0170  C5 9E 00517          MOVAB   P.ADC, CMD_DESC+4                      ;
                6E              02 D0 0051D          MOVL    #2, ATTRIBUTES                         ;
                                53 DD 00520          PUSHL   SET_PTR                                ;
                          18    AE 9F 00522          PUSHAB  CMD_DESC                               ;
                          08    AE 9F 00525          PUSHAB  ATTRIBUTES                             ;
                                54 DD 00528          PUSHL   IF_PTR                                 ;
                          14    AE 9F 0052A          PUSHAB  KEY_DESC                               ;
                                56 DD 0052D          PUSHL   R6                                     ;
                67              06 FB 0052F          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                52              50 D0 00532          MOVL    R0, STATUS                             ;
                05              52 E8 00535          BLBS    STATUS, 23$                            ;
                                52 DD 00538          PUSHL   STATUS                                 ;
```

```
                68              01 FB 0053A             CALLS   #1, LIB$SIGNAL
                54        1C    AE 9E 0053D 23$:        MOVAB   BLUE_DESC, IF_PTR                       ;0517
                                53 D4 00541             CLRL    SET_PTR
          04    AE              03 B0 00543             MOVW    #3, KEY_DESC
          08    AE        017A  C5 9E 00547             MOVAB   P.ADD, KEY_DESC+4
          14    AE              0C B0 0054D             MOVW    #12, CMD_DESC
          18    AE        017D  C5 9E 00551             MOVAB   P.ADE, CMD_DESC+4
                6E              02 D0 00557             MOVL    #2, ATTRIBUTES
                                53 DD 0055A             PUSHL   SET_PTR
                          18    AE 9F 0055C             PUSHAB  CMD_DESC
                          08    AE 9F 0055F             PUSHAB  ATTRIBUTES
                                54 DD 00562             PUSHL   IF_PTR
                          14    AE 9F 00564             PUSHAB  KEY_DESC
                                56 DD 00567             PUSHL   R6
                67              06 FB 00569             CALLS   #6, SMG$ADD_KEY_DEF
                52              50 D0 0056C             MOVL    R0, STATUS
                05              52 E8 0056F             BLBS    STATUS, 24$
                                52 DD 00572             PUSHL   STATUS
                68              01 FB 00574             CALLS   #1, LIB$SIGNAL
                                53 7C 00577 24$:        CLRQ    SET_PTR                                 ;0518
          04    AE              03 B0 00579             MOVW    #3, KEY_DESC
          08    AE        0189  C5 9E 0057D             MOVAB   P.ADF, KEY_DESC+4
          14    AE              0C B0 00583             MOVW    #12, CMD_DESC
          18    AE        018C  C5 9E 00587             MOVAB   P.ADG, CMD_DESC+4
                6E              03 D0 0058D             MOVL    #3, ATTRIBUTES
                                53 DD 00590             PUSHL   SET_PTR
                          18    AE 9F 00592             PUSHAB  CMD_DESC
                          08    AE 9F 00595             PUSHAB  ATTRIBUTES
                                54 DD 00598             PUSHL   IF_PTR
                          14    AE 9F 0059A             PUSHAB  KEY_DESC
                                56 DD 0059D             PUSHL   R6
                67              06 FB 0059F             CALLS   #6, SMG$ADD_KEY_DEF
                52              50 D0 005A2             MOVL    R0, STATUS
                05              52 E8 005A5             BLBS    STATUS, 25$
                                52 DD 005A8             PUSHL   STATUS
                68              01 FB 005AA             CALLS   #1, LIB$SIGNAL
                54        1C    AE 9E 005AD 25$:        MOVAB   BLUE_DESC, IF_PTR                       ;0519
                                53 D4 005B1             CLRL    SET_PTR
          04    AE              03 B0 005B3             MOVW    #3, KEY_DESC
          08    AE        0198  C5 9E 005B7             MOVAB   P.ADH, KEY_DESC+4
          14    AE              0D B0 005BD             MOVW    #13, CMD_DESC
          18    AE        019B  C5 9E 005C1             MOVAB   P.ADI, CMD_DESC+4
                                6E D4 005C7             CLRL    ATTRIBUTES
                                53 DD 005C9             PUSHL   SET_PTR
                          18    AE 9F 005CB             PUSHAB  CMD_DESC
                          08    AE 9F 005CE             PUSHAB  ATTRIBUTES
                                54 DD 005D1             PUSHL   IF_PTR
                          14    AE 9F 005D3             PUSHAB  KEY_DESC
                                56 DD 005D6             PUSHL   R6
                67              06 FB 005D8             CALLS   #6, SMG$ADD_KEY_DEF
                52              50 D0 005DB             MOVL    R0, STATUS
                05              52 E8 005DE             BLBS    STATUS, 26$
                                52 DD 005E1             PUSHL   STATUS
                68              01 FB 005E3             CALLS   #1, LIB$SIGNAL
                                53 7C 005E6 26$:        CLRQ    SET_PTR                                 ;0520
          04    AE              05 B0 005E8             MOVW    #5, KEY_DESC
          08    AE        01A8  C5 9E 005EC             MOVAB   P.ADJ, KEY_DESC+4
```

```
                14   AE          02   B0 005F2          MOVW    #2, CMD_DESC                                    ;
                18   AE    01AD  C5   9E 005F6          MOVAB   P.ADK, CMD_DESC+4                               ;
                     6E          02   D0 005FC          MOVL    #2, ATTRIBUTES                                  ;
                                 53   DD 005FF          PUSHL   SET_PTR                                         ;
                                 18   AE 9F 00601       PUSHAB  CMD_DESC                                        ;
                                 08   AE 9F 00604       PUSHAB  ATTRIBUTES                                      ;
                                 54   DD 00607          PUSHL   IF_PTR                                          ;
                                 14   AE 9F 00609       PUSHAB  KEY_DESC                                        ;
                                 56   DD 0060C          PUSHL   R6                                              ;
                           67         06   FB 0060E     CALLS   #6, SMG$ADD_KEY_DEF                             ;
                           52         50   D0 00611     MOVL    R0, STATUS                                      ;
                           05         52   E8 00614     BLBS    STATUS, 27$                                     ;
                                      52   DD 00617     PUSHL   STATUS                                          ;
                           68         01   FB 00619     CALLS   #1, LIB$SIGNAL                                  ;
                                      53   7C 0061C 27$: CLRQ    SET_PTR                                         ; 0525
                04   AE               03   B0 0061E     MOVW    #3, KEY_DESC                                    ;
                08   AE    01AF       C5   9E 00622     MOVAB   P.ADL, KEY_DESC+4                               ;
                14   AE               07   B0 00628     MOVW    #7, CMD_DESC                                    ;
                18   AE    01B2       C5   9E 0062C     MOVAB   P.ADM, CMD_DESC+4                               ;
                     6E          02   D0 00632          MOVL    #2, ATTRIBUTES                                  ;
                                 53   DD 00635          PUSHL   SET_PTR                                         ;
                                 18   AE 9F 00637       PUSHAB  CMD_DESC                                        ;
                                 08   AE 9F 0063A       PUSHAB  ATTRIBUTES                                      ;
                                 54   DD 0063D          PUSHL   IF_PTR                                          ;
                                 14   AE 9F 0063F       PUSHAB  KEY_DESC                                        ;
                                 56   DD 00642          PUSHL   R6                                              ;
                           67         06   FB 00644     CALLS   #6, SMG$ADD_KEY_DEF                             ;
                           52         50   D0 00647     MOVL    R0, STATUS                                      ;
                           05         52   E8 0064A     BLBS    STATUS, 28$                                     ;
                                      52   DD 0064D     PUSHL   STATUS                                          ;
                           68         01   FB 0064F     CALLS   #1, LIB$SIGNAL                                  ;
                                      53   7C 00652 28$: CLRQ    SET_PTR                                         ; 0526
                04   AE               03   B0 00654     MOVW    #3, KEY_DESC                                    ;
                08   AE    01B9       C5   9E 00658     MOVAB   P.ADN, KEY_DESC+4                               ;
                14   AE               0B   B0 0065E     MOVW    #11, CMD_DESC                                   ;
                18   AE    01BC       C5   9E 00662     MOVAB   P.ADO, CMD_DESC+4                               ;
                     6E          03   D0 00668          MOVL    #3, ATTRIBUTES                                  ;
                                 53   DD 0066B          PUSHL   SET_PTR                                         ;
                                 18   AE 9F 0066D       PUSHAB  CMD_DESC                                        ;
                                 08   AE 9F 00670       PUSHAB  ATTRIBUTES                                      ;
                                 54   DD 00673          PUSHL   IF_PTR                                          ;
                                 14   AE 9F 00675       PUSHAB  KEY_DESC                                        ;
                                 56   DD 00678          PUSHL   R6                                              ;
                           67         06   FB 0067A     CALLS   #6, SMG$ADD_KEY_DEF                             ;
                           52         50   D0 0067D     MOVL    R0, STATUS                                      ;
                           05         52   E8 00680     BLBS    STATUS, 29$                                     ;
                                      52   DD 00683     PUSHL   STATUS                                          ;
                           68         01   FB 00685     CALLS   #1, LIB$SIGNAL                                  ;
                           54    0C   AE   9E 00688 29$: MOVAB   GOLD_DESC, IF_PTR                              ; 0527
                                      53   D4 0068C     CLRL    SET_PTR                                         ;
                04   AE               03   B0 0068E     MOVW    #3, KEY_DESC                                    ;
                08   AE    01C7       C5   9E 00692     MOVAB   P.ADP, KEY_DESC+4                               ;
                14   AE               0D   B0 00698     MOVW    #13, CMD_DESC                                   ;
                18   AE    01CA       C5   9E 0069C     MOVAB   P.ADQ, CMD_DESC+4                               ;
                     6E          03   D0 006A2          MOVL    #3, ATTRIBUTES                                  ;
                                 53   DD 006A5          PUSHL   SET_PTR                                         ;
                                 18   AE 9F 006A7       PUSHAB  CMD_DESC                                        ;
```

```
                08 AE 9F 006AA          PUSHAB  ATTRIBUTES                                      ;
                   54 DD 006AD          PUSHL   IF_PTR                                          ;
                14 AE 9F 006AF          PUSHAB  KEY_DESC                                        ;
                   56 DD 006B2          PUSHL   R6                                              ;
             67    06 FB 006B4          CALLS   #6, SMG$ADD_KEY_DEF                             ;
             52    50 D0 006B7          MOVL    R0, STATUS                                      ;
             05    52 E8 006BA          BLBS    STATUS, 30$                                     ;
                   52 DD 006BD          PUSHL   STATUS                                          ;
             68    01 FB 006BF          CALLS   #1, LIB$SIGNAL                                  ;
             54    1C AE 9E 006C2 30$:  MOVAB   BLUE_DESC, IF_PTR                               ; 0528
                   53 D4 006C6          CLRL    SET_PTR                                         ;
          04 AE    03 B0 006C8          MOVW    #3, KEY_DESC                                    ;
          08 AE 01D7 C5 9E 006CC        MOVAB   P.ADR, KEY_DESC+4                               ;
          14 AE    0C B0 006D2          MOVW    #12, CMD_DESC                                   ;
          18 AE 01DA C5 9E 006D6        MOVAB   P.ADS, CMD_DESC+4                               ;
                   6E D4 006DC          CLRL    ATTRIBUTES                                      ;
                   53 DD 006DE          PUSHL   SET_PTR                                         ;
                18 AE 9F 006E0          PUSHAB  CMD_DESC                                        ;
                08 AE 9F 006E3          PUSHAB  ATTRIBUTES                                      ;
                   54 DD 006E6          PUSHL   IF_PTR                                          ;
                14 AE 9F 006E8          PUSHAB  KEY_DESC                                        ;
                   56 DD 006EB          PUSHL   R6                                              ;
             67    06 FB 006ED          CALLS   #6, SMG$ADD_KEY_DEF                             ;
             52    50 D0 006F0          MOVL    R0, STATUS                                      ;
             05    52 E8 006F3          BLBS    STATUS, 31$                                     ;
                   52 DD 006F6          PUSHL   STATUS                                          ;
             68    01 FB 006F8          CALLS   #1, LIB$SIGNAL                                  ;
                   53 7C 006FB 31$:     CLRQ    SET_PTR                                         ; 0529
          04 AE    03 B0 006FD          MOVW    #3, KEY_DESC                                    ;
          08 AE 01E6 C5 9E 00701        MOVAB   P.ADT, KEY_DESC+4                               ;
          14 AE    19 B0 00707          MOVW    #25, CMD_DESC                                   ;
          18 AE 01E9 C5 9E 0070B        MOVAB   P.ADU, CMD_DESC+4                               ;
             6E    03 D0 00711          MOVL    #3, ATTRIBUTES                                  ;
                   53 DD 00714          PUSHL   SET_PTR                                         ;
                18 AE 9F 00716          PUSHAB  CMD_DESC                                        ;
                08 AE 9F 00719          PUSHAB  ATTRIBUTES                                      ;
                   54 DD 0071C          PUSHL   IF_PTR                                          ;
                14 AE 9F 0071E          PUSHAB  KEY_DESC                                        ;
                   56 DD 00721          PUSHL   R6                                              ;
             67    06 FB 00723          CALLS   #6, SMG$ADD_KEY_DEF                             ;
             52    50 D0 00726          MOVL    R0, STATUS                                      ;
             05    52 E8 00729          BLBS    STATUS, 32$                                     ;
                   52 DD 0072C          PUSHL   STATUS                                          ;
             68    01 FB 0072E          CALLS   #1, LIB$SIGNAL                                  ;
             54    0C AE 9E 00731 32$:  MOVAB   GOLD_DESC, IF_PTR                               ; 0530
                   53 D4 00735          CLRL    SET_PTR                                         ;
          04 AE    03 B0 00737          MOVW    #3, KEY_DESC                                    ;
          08 AE 0202 C5 9E 0073B        MOVAB   P.ADV, KEY_DESC+4                               ;
          14 AE    19 B0 00741          MOVW    #25, CMD_DESC                                   ;
          18 AE 0205 C5 9E 00745        MOVAB   P.ADW, CMD_DESC+4                               ;
             6E    03 D0 0074B          MOVL    #3, ATTRIBUTES                                  ;
                   53 DD 0074E          PUSHL   SET_PTR                                         ;
                18 AE 9F 00750          PUSHAB  CMD_DESC                                        ;
                08 AE 9F 00753          PUSHAB  ATTRIBUTES                                      ;
                   54 DD 00756          PUSHL   IF_PTR                                          ;
                14 AE 9F 00758          PUSHAB  KEY_DESC                                        ;
                   56 DD 0075B          PUSHL   R6                                              ;
```

```
                67              06  FB 0075D          CALLS   #6, SMG$ADD_KEY_DEF
                52              50  D0 00760          MOVL    R0, STATUS
                05              52  E8 00763          BLBS    STATUS, 33$
                                52  DD 00766          PUSHL   STATUS
                68              01  FB 00768          CALLS   #1, LIB$SIGNAL
                54        1C    AE  9E 0076B 33$:     MOVAB   BLUE_DESC, IF_PTR                   ; 0531
                                53  D4 0076F          CLRL    SET_PTR
          04    AE              03  B0 00771          MOVW    #3, KEY_DESC
          08    AE        021E  C5  9E 00775          MOVAB   P.ADX, KEY_DESC+4
          14    AE              19  B0 00778          MOVW    #25, CMD_DESC
          18    AE        0221  C5  9E 0077F          MOVAB   P.ADY, CMD_DESC+4
                6E              03  D0 00785          MOVL    #3, ATTRIBUTES
                                53  DD 00788          PUSHL   SET_PTR
                          18    AE  9F 0078A          PUSHAB  CMD_DESC
                          08    AE  9F 0078D          PUSHAB  ATTRIBUTES
                                54  DD 00790          PUSHL   IF_PTR
                          14    AE  9F 00792          PUSHAB  KEY_DESC
                                56  DD 00795          PUSHL   R6
                67              06  FB 00797          CALLS   #6, SMG$ADD_KEY_DEF
                52              50  D0 0079A          MOVL    R0, STATUS
                05              52  E8 0079D          BLBS    STATUS, 34$
                                52  DD 007A0          PUSHL   STATUS
                68              01  FB 007A2          CALLS   #1, LIB$SIGNAL
                                53  7C 007A5 34$:     CLRQ    SET_PTR                             ; 0536
          04    AE              03  B0 007A7          MOVW    #3, KEY_DESC
          08    AE        023A  C5  9E 007AB          MOVAB   P.ADZ, KEY_DESC+4
          14    AE              04  B0 007B1          MOVW    #4, CMD_DESC
          18    AE        023D  C5  9E 007B5          MOVAB   P.AEA, CMD_DESC+4
                6E              02  D0 007BB          MOVL    #2, ATTRIBUTES
                                53  DD 007BE          PUSHL   SET_PTR
                          18    AE  9F 007C0          PUSHAB  CMD_DESC
                          08    AE  9F 007C3          PUSHAB  ATTRIBUTES
                                54  DD 007C6          PUSHL   IF_PTR
                          14    AE  9F 007C8          PUSHAB  KEY_DESC
                                56  DD 007CB          PUSHL   R6
                67              06  FB 007CD          CALLS   #6, SMG$ADD_KEY_DEF
                52              50  D0 007D0          MOVL    R0, STATUS
                05              52  E8 007D3          BLBS    STATUS, 35$
                                52  DD 007D6          PUSHL   STATUS
                68              01  FB 007D8          CALLS   #1, LIB$SIGNAL
                54        0C    AE  9E 007DB 35$:     MOVAB   GOLD_DESC, IF_PTR                   ; 0537
                                53  D4 007DF          CLRL    SET_PTR
          04    AE              03  B0 007E1          MOVW    #3, KEY_DESC
          08    AE        0241  C5  9E 007E5          MOVAB   P.AEB, KEY_DESC+4
          14    AE              09  B0 007EB          MOVW    #9, CMD_DESC
          18    AE        0244  C5  9E 007EF          MOVAB   P.AEC, CMD_DESC+4
                6E              02  D0 007F5          MOVL    #2, ATTRIBUTES
                                53  DD 007F8          PUSHL   SET_PTR
                          18    AE  9F 007FA          PUSHAB  CMD_DESC
                          08    AE  9F 007FD          PUSHAB  ATTRIBUTES
                                54  DD 00800          PUSHL   IF_PTR
                          14    AE  9F 00802          PUSHAB  KEY_DESC
                                56  DD 00805          PUSHL   R6
                67              06  FB 00807          CALLS   #6, SMG$ADD_KEY_DEF
                52              50  D0 0080A          MOVL    R0, STATUS
                05              52  E8 0080D          BLBS    STATUS, 36$
                                52  DD 00810          PUSHL   STATUS
```

```
                68              01 FB 00812          CALLS   #1, LIB$SIGNAL
                54        1C    AE 9E 00815 36$:     MOVAB   BLUE_DESC, IF_PTR                0538
                                53 D4 00819          CLRL    SET_PTR
          04    AE              03 B0 0081B          MOVW    #3, KEY_DESC
          08    AE        024D  C5 9E 0081F          MOVAB   P.AED, KEY_DESC+4
          14    AE              09 B0 00825          MOVW    #9, CMD_DESC
          18    AE        0250  C5 9E 00829          MOVAB   P.AEE, CMD_DESC+4
                6E              02 D0 0082F          MOVL    #2, ATTRIBUTES
                                53 DD 00832          PUSHL   SET_PTR
                          18    AE 9F 00834          PUSHAB  CMD_DESC
                          08    AE 9F 00837          PUSHAB  ATTRIBUTES
                                54 DD 0083A          PUSHL   IF_PTR
                          14    AE 9F 0083C          PUSHAB  KEY_DESC
                                56 DD 0083F          PUSHL   R6
                67              06 FB 00841          CALLS   #6, SMG$ADD_KEY_DEF
                52              50 D0 00844          MOVL    R0, STATUS
                05              52 E8 00847          BLBS    STATUS, 37$
                                52 DD 0084A          PUSHL   STATUS
                68              01 FB 0084C          CALLS   #1, LIB$SIGNAL
                                53 7C 0084F 37$:     CLRQ    SET_PTR                          0539
          04    AE              06 B0 00851          MOVW    #6, KEY_DESC
          08    AE        0259  C5 9E 00855          MOVAB   P.AEF, KEY_DESC+4
                            14  AE B4 0085B          CLRW    CMD_DESC
          18    AE        025F  C5 9E 0085E          MOVAB   P.AEG, CMD_DESC+4
                6E              D4 00864             CLRL    ATTRIBUTES
                                53 DD 00866          PUSHL   SET_PTR
                          18    AE 9F 00868          PUSHAB  CMD_DESC
                          08    AE 9F 0086B          PUSHAB  ATTRIBUTES
                                54 DD 0086E          PUSHL   IF_PTR
                          14    AE 9F 00870          PUSHAB  KEY_DESC
                                56 DD 00873          PUSHL   R6
                67              06 FB 00875          CALLS   #6, SMG$ADD_KEY_DEF
                52              50 D0 00878          MOVL    R0, STATUS
                05              52 E8 0087B          BLBS    STATUS, 38$
                                52 DD 0087E          PUSHL   STATUS
                68              01 FB 00880          CALLS   #1, LIB$SIGNAL
                54        0C    AE 9E 00883 38$:     MOVAB   GOLD_DESC, IF_PTR                0540
                                53 D4 00887          CLRL    SET_PTR
          04    AE              06 B0 00889          MOVW    #6, KEY_DESC
          08    AE        025F  C5 9E 0088D          MOVAB   P.AEH, KEY_DESC+4
                            14  AE B4 00893          CLRW    CMD_DESC
          18    AE        0265  C5 9E 00896          MOVAB   P.AEI, CMD_DESC+4
                6E              D4 0089C             CLRL    ATTRIBUTES
                                53 DD 0089E          PUSHL   SET_PTR
                          18    AE 9F 008A0          PUSHAB  CMD_DESC
                          08    AE 9F 008A3          PUSHAB  ATTRIBUTES
                                54 DD 008A6          PUSHL   IF_PTR
                          14    AE 9F 008A8          PUSHAB  KEY_DESC
                                56 DD 008AB          PUSHL   R6
                67              06 FB 008AD          CALLS   #6, SMG$ADD_KEY_DEF
                52              50 D0 008B0          MOVL    R0, STATUS
                05              52 E8 008B3          BLBS    STATUS, 39$
                                52 DD 008B6          PUSHL   STATUS
                68              01 FB 008B8          CALLS   #1, LIB$SIGNAL
                54        1C    AE 9E 008BB 39$:     MOVAB   BLUE_DESC, IF_PTR                0541
                                53 D4 008BF          CLRL    SET_PTR
          04    AE              06 B0 008C1          MOVW    #6, KEY_DESC
```

```
                08   AE      0235    C5  9E 008C5          MOVAB   P.AEJ, KEY_DESC+4                ;
                     14      AE      B4 008CB           CLRW    CMD_DESC                        ;
                18   AE      026B    C5  9E 008CE           MOVAB   P.AEK, CMD_DESC+4               ;
                                     6E  D4 008D4           CLRL    ATTRIBUTES                      ;
                                     53  DD 008D6           PUSHL   SET_PTR                         ;
                             18      AE  9F 008D8           PUSHAB  CMD_DESC                        ;
                             08      AE  9F 008DB           PUSHAB  ATTRIBUTES                      ;
                                     54  DD 008DE           PUSHL   IF_PTR                          ;
                             14      AE  9F 008E0           PUSHAB  KEY_DESC                        ;
                                     56  DD 008E3           PUSHL   R6                              ;
                         67              06  FB 008E5           CALLS   #6, SMG$ADD_KEY_DEF             ;
                         52              50  D0 008E8           MOVL    R0, STATUS                      ;
                         05              52  E8 008EB           BLBS    STATUS, 40$                     ;
                                         52  DD 008EE           PUSHL   STATUS                          ;
                         66              01  FB 008F0           CALLS   #1, LIB$SIGNAL                  ;
                                         53  7C 008F3 40$:      CLRQ    SET_PTR                         ; 0546
                04   AE                  05  B0 008F5           MOVW    #5, KEY_DESC                    ;
                08   AE      026B        C5  9E 008F9           MOVAB   P.AEL, KEY_DESC+4               ;
                14   AE                  0F  B0 008FF           MOVW    #15, CMD_DESC                   ;
                18   AE      0270        C5  9E 00903           MOVAB   P.AEM, CMD_DESC+4               ;
                     6E                  03  D0 00909           MOVL    #3, ATTRIBUTES                  ;
                                         53  DD 0090C           PUSHL   SET_PTR                         ;
                             18      AE  9F 0090E           PUSHAB  CMD_DESC                        ;
                             08      AE  9F 00911           PUSHAB  ATTRIBUTES                      ;
                                     54  DD 00914           PUSHL   IF_PTR                          ;
                             14      AE  9F 00916           PUSHAB  KEY_DESC                        ;
                                     56  DD 00919           PUSHL   R6                              ;
                         67              06  FB 0091B           CALLS   #6, SMG$ADD_KEY_DEF             ;
                         52              50  D0 0091E           MOVL    R0, STATUS                      ;
                         05              52  E8 00921           BLBS    STATUS, 41$                     ;
                                         52  DD 00924           PUSHL   STATUS                          ;
                         68              01  FB 00926           CALLS   #1, LIB$SIGNAL                  ;
                         54      0C      AE  9E 00929 41$:      MOVAB   GOLD_DESC, IF_PTR               ; 0547
                                         53  D4 0092D           CLRL    SET_PTR                         ;
                04   AE                  05  B0 0092F           MOVW    #5, KEY_DESC                    ;
                08   AE      027F        C5  9E 00933           MOVAB   P.AEN, KEY_DESC+4               ;
                14   AE                  0F  B0 00939           MOVW    #15, CMD_DESC                   ;
                18   AE      0284        C5  9E 0093D           MOVAB   P.AEO, CMD_DESC+4               ;
                     6E                  03  D0 00943           MOVL    #3, ATTRIBUTES                  ;
                                         53  DD 00946           PUSHL   SET_PTR                         ;
                             18      AE  9F 00948           PUSHAB  CMD_DESC                        ;
                             08      AE  9F 0094B           PUSHAB  ATTRIBUTES                      ;
                                     54  DD 0094E           PUSHL   IF_PTR                          ;
                             14      AE  9F 00950           PUSHAB  KEY_DESC                        ;
                                     56  DD 00953           PUSHL   R6                              ;
                         67              06  FB 00955           CALLS   #6, SMG$ADD_KEY_DEF             ;
                         52              50  D0 00958           MOVL    R0, STATUS                      ;
                         05              52  E8 0095B           BLBS    STATUS, 42$                     ;
                                         52  DD 0095E           PUSHL   STATUS                          ;
                         68              01  FB 00960           CALLS   #1, LIB$SIGNAL                  ;
                         54      1C      AE  9E 00963 42$:      MOVAB   BLUE_DESC, IF_PTR               ; 0548
                                         53  D4 00967           CLRL    SET_PTR                         ;
                04   AE                  05  B0 00969           MOVW    #5, KEY_DESC                    ;
                08   AE      0293        C5  9E 0096D           MOVAB   P.AEP, KEY_DESC+4               ;
                14   AE                  0F  B0 00973           MOVW    #15, CMD_DESC                   ;
                18   AE      0298        C5  9E 00977           MOVAB   P.AEQ, CMD_DESC+4               ;
                     6E                  03  D0 0097D           MOVL    #3, ATTRIBUTES                  ;
```

```
                        53   DD 00980          PUSHL   SET_PTR                                    ;
                     18 AE   9F 00982          PUSHAB  CMD_DESC                                   ;
                     08 AE   9F 00985          PUSHAB  ATTRIBUTES                                 ;
                        54   DD 00988          PUSHL   IF_PTR                                     ;
                     14 AE   9F 0098A          PUSHAB  KEY_DESC                                   ;
                        56   DD 0098D          PUSHL   R6                                         ;
                  67        06   FB 0098F          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                  52        50   D0 00992          MOVL    R0, STATUS                             ;
                  05        52   E8 00995          BLBS    STATUS, 43$                            ;
                            52   DD 00998          PUSHL   STATUS                                 ;
                  68        01   FB 0099A          CALLS   #1, LIB$SIGNAL                         ;
                            53   7C 0099D 43$:      CLRQ    SET_PTR                                ;  0554
04 AE                       04   B0 0099F          MOVW    #4, KEY_DESC                           ;
08 AE     02A7              C5   9E 009A3          MOVAB   P.AER, KEY_DESC+4                      ;
14 AE                       13   B0 009A9          MOVW    #19, CMD_DESC                          ;
18 AE     02AB              C5   9E 009AD          MOVAB   P.AES, CMD_DESC+4                      ;
                  6E        02   D0 009B3          MOVL    #2, ATTRIBUTES                         ;
                            53   DD 009B6          PUSHL   SET_PTR                                ;
                     18 AE   9F 009B8          PUSHAB  CMD_DESC                                   ;
                     08 AE   9F 009BB          PUSHAB  ATTRIBUTES                                 ;
                        54   DD 009BE          PUSHL   IF_PTR                                     ;
                     14 AE   9F 009C0          PUSHAB  KEY_DESC                                   ;
                        56   DD 009C3          PUSHL   R6                                         ;
                  67        06   FB 009C5          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                  52        50   D0 009C8          MOVL    R0, STATUS                             ;
                  05        52   E8 009CB          BLBS    STATUS, 44$                            ;
                            52   DD 009CE          PUSHL   STATUS                                 ;
                  68        01   FB 009D0          CALLS   #1, LIB$SIGNAL                         ;
                            53   7C 009D3 44$:      CLRQ    SET_PTR                                ;  0555
04 AE                       0B   B0 009D5          MOVW    #11, KEY_DESC                          ;
08 AE     02BE              C5   9E 009D9          MOVAB   P.AET, KEY_DESC+4                      ;
14 AE                       09   B0 009DF          MOVW    #9, CMD_DESC                           ;
18 AE     02C9              C5   9E 009E3          MOVAB   P.AEU, CMD_DESC+4                      ;
                  6E        03   D0 009E9          MOVL    #3, ATTRIBUTES                         ;
                            53   DD 009EC          PUSHL   SET_PTR                                ;
                     18 AE   9F 009EE          PUSHAB  CMD_DESC                                   ;
                     08 AE   9F 009F1          PUSHAB  ATTRIBUTES                                 ;
                        54   DD 009F4          PUSHL   IF_PTR                                     ;
                     14 AE   9F 009F6          PUSHAB  KEY_DESC                                   ;
                        56   DD 009F9          PUSHL   R6                                         ;
                  67        06   FB 009FB          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                  52        50   D0 009FE          MOVL    R0, STATUS                             ;
                  05        52   E8 00A01          BLBS    STATUS, 45$                            ;
                            52   DD 00A04          PUSHL   STATUS                                 ;
                  68        01   FB 00A06          CALLS   #1, LIB$SIGNAL                         ;
                            53   7C 00A09 45$:      CLRQ    SET_PTR                                ;  0556
04 AE                       0B   B0 00A0B          MOVW    #11, KEY_DESC                          ;
08 AE     02D2              C5   9E 00A0F          MOVAB   P.AEV, KEY_DESC+4                      ;
14 AE                       0B   B0 00A15          MOVW    #11, CMD_DESC                          ;
18 AE     02DD              C5   9E 00A19          MOVAB   P.AEW, CMD_DESC+4                      ;
                  6E        03   D0 00A1F          MOVL    #3, ATTRIBUTES                         ;
                            53   DD 00A22          PUSHL   SET_PTR                                ;
                     18 AE   9F 00A24          PUSHAB  CMD_DESC                                   ;
                     08 AE   9F 00A27          PUSHAB  ATTRIBUTES                                 ;
                        54   DD 00A2A          PUSHL   IF_PTR                                     ;
                     14 AE   9F 00A2C          PUSHAB  KEY_DESC                                   ;
                        56   DD 00A2F          PUSHL   R6                                         ;
```

```
                67              06  FB 00A31          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                52              50  D0 00A34          MOVL    R0, STATUS                             ;
                05              52  E8 00A37          BLBS    STATUS, 46$                            ;
                                52  DD 00A3A          PUSHL   STATUS                                 ;
                68              01  FB 00A3C          CALLS   #1, LIB$SIGNAL                         ;
                                53  7C 00A3F 46$:     CLRQ    SET_PTR                                ; 0557
            04  AE              06  B0 00A41          MOVW    #6, KEY_DESC                           ;
            08  AE      02E8    C5  9E 00A45          MOVAB   P.AEX, KEY_DESC+4                      ;
            14  AE              19  B0 00A4B          MOVW    #25, CMD_DESC                          ;
            18  AE      02EE    C5  9E 00A4F          MOVAB   P.AEY, CMD_DESC+4                      ;
                6E              03  D0 00A55          MOVL    #3, ATTRIBUTES                         ;
                                53  DD 00A58          PUSHL   SET_PTR                                ;
                        18      AE  9F 00A5A          PUSHAB  CMD_DESC                               ;
                        08      AE  9F 00A5D          PUSHAB  ATTRIBUTES                             ;
                                54  DD 00A60          PUSHL   IF_PTR                                 ;
                        14      AE  9F 00A62          PUSHAB  KEY_DESC                               ;
                                56  DD 00A65          PUSHL   R6                                     ;
                67              06  FB 00A67          CALLS   #6, SMG$ADD_KEY_DEF                    ;
                52              50  D0 00A6A          MOVL    R0, STATUS                             ;
                05              52  E8 00A6D          BLBS    STATUS, 47$                            ;
                                52  DD 00A70          PUSHL   STATUS                                 ;
                68              01  FB 00A72          CALLS   #1, LIB$SIGNAL                         ;
                                    04 00A75 47$:     RET                                            ; 0564

; Routine Size:  2678 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  436     0565  1 ROUTINE DBG$SCR_CREATE_DISPLAY(NAMEPTR, KIND, WPTR, SIZE, CONTENT_PTR) =
;  437     0566  1 !
;  438     0567  1 ! FUNCTION
;  439     0568  1 !       This routine creates a new screen display.  It accepts the name of
;  440     0569  1 !       the new display and its initial attributes as input.  If a display
;  441     0570  1 !       by the same name already exists, an error is signalled, but other-
;  442     0571  1 !       wise this routine creates a Screen Display Entry for the new display
;  443     0572  1 !       and fills the display name and the other attributes into the Display
;  444     0573  1 !       Entry.  The new display is then appended to the Screen Display List.
;  445     0574  1 !
;  446     0575  1 ! INPUTS
;  447     0576  1 !       NAMEPTR - A pointer to the name of the display to be created.  The
;  448     0577  1 !                 name is represented as a Counted ASCII string.
;  449     0578  1 !
;  450     0579  1 !       KIND    - The kind of display to be created.  This basically refers
;  451     0580  1 !                 to how the contents are generated: it may be NORMAL, SOURCE,
;  452     0581  1 !                 DO, etc.
;  453     0582  1 !
;  454     0583  1 !       WPTR    - A pointer to a Screen Window Entry.  This Window Entry
;  455     0584  1 !                 defines the initial screen window for the new display.
;  456     0585  1 !
;  457     0586  1 !       SIZE    - The maximum size of the display in text lines.  If more
;  458     0587  1 !                 than SIZE lines of text are later generated for this
;  459     0588  1 !                 display, the oldest lines are discarded so that at most
;  460     0589  1 !                 SIZE lines are kept at any one time.
;  461     0590  1 !
;  462     0591  1 !       CONTENT_PTR - A pointer to the DEBUG command list which determines
;  463     0592  1 !                 the contents of this display if this is an automatically
;  464     0593  1 !                 updated display.  If this is not an automatically updated
;  465     0594  1 !                 display according to the KIND parameter, then CONTENT_PTR
;  466     0595  1 !                 is ignored and should be zero.
;  467     0596  1 !
;  468     0597  1 ! OUTPUTS
;  469     0598  1 !       A pointer to the created Screen Display Entry is returned as this
;  470     0599  1 !                 routine's value.
;  471     0600  1 !
;  472     0601  1
;  473     0602  2     BEGIN
;  474     0603  2
;  475     0604  2     MAP
;  476     0605  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to ASCIC display name
;  477     0606  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to Screen Window Entry
;  478     0607  2
;  479     0608  2     LOCAL
;  480     0609  2         BLINK: REF DBG$DISP_ENTRY,      ! Pointer to last Screen Display Entry
;  481     0610  2                                         !      on the Screen Display List
;  482     0611  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to new Screen Display Entry
;  483     0612  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to label Display Line Entry
;  484     0613  2         FLINK: REF DBG$DISP_ENTRY;      ! Pointer to the list head for the
;  485     0614  2                                         !      Screen Display List
;  486     0615  2
;  487     0616  2
;  488     0617  2
;  489     0618  2     ! See if a display by the specified name already exists.  If it does, we
;  490     0619  2     ! signal an error and do not allow the new display to be created.
;  491     0620  2     !
;  492     0621  2     DISPTR = DBG$SCR_LOOKUP_DISPLAY(.NAMEPTR);
```

```
;  493          0622  2      IF .DISPTR NEQ 0 THEN SIGNAL(DBG$_DISPEXISTS, 1, .NAMEPTR);
;  494          0623  2
;  495          0624  2
;  496          0625  2      ! Allocate a memory block for the new Screen Display Entry and fill in all
;  497          0626  2      ! the relevant fields of that entry.
;  498          0627  2      !
;  499          0628  2      DISPTR = DBG$GET_MEMORY(DBG$K_DISP_ENTSIZE + (.NAMEPTR[0] + %UPVAL)/%UPVAL);
;  500          0629  2      DISPTR[DBG$B_DISP_KIND] = .KIND;
;  501          0630  2      DISPTR[DBG$W_DISP_RBEG] = .WPTR[DBG$W_WINDOW_RBEG];
;  502          0631  2      DISPTR[DBG$W_DISP_RLEN] = .WPTR[DBG$W_WINDOW_RLEN];
;  503          0632  2      DISPTR[DBG$W_DISP_CBEG] = .WPTR[DBG$W_WINDOW_CBEG];
;  504          0633  2      DISPTR[DBG$W_DISP_CLEN] = .WPTR[DBG$W_WINDOW_CLEN];
;  505          0634  2      DISPTR[DBG$W_DISP_DROW] = 1;
;  506          0635  2      DISPTR[DBG$W_DISP_DCOL] = 1;
;  507          0636  2      DISPTR[DBG$W_DISP_MAX_LINECNT] = MAX(1, .SIZE);
;  508          0637  2      DISPTR[DBG$V_DISP_NEWDISP] = TRUE;
;  509          0638  2      DISPTR[DBG$V_DISP_INVSCR] = TRUE;
;  510          0639  2      DISPTR[DBG$L_DISP_MARKLINE] = -1;
;  511          0640  2      DISPTR[DBG$L_DISP_CMDLIST] = .CONTENT_PTR;
;  512          0641  2      CH$MOVE(.NAMEPTR[0] + 1, NAMEPTR[0], DISPTR[DBG$A_DISP_NAME]);
;  513          0642  2
;  514          0643  2
;  515          0644  2      ! Initialize the list head for the list of Display Line Entries.  This
;  516          0645  2      ! sets up an empty Line Entry list.
;  517          0646  2      !
;  518          0647  2      DISPTR[DBG$L_DISP_START_LINE_PTR] = DISPTR[DBG$L_DISP_START_LINE_PTR];
;  519          0648  2      DISPTR[DBG$L_DISP_END_LINE_PTR] = DISPTR[DBG$L_DISP_START_LINE_PTR];
;  520          0649  2
;  521          0650  2
;  522          0651  2      ! Link the new Screen Display Entry into the Screen Display List.  The new
;  523          0652  2      ! entry is linked in at the end of the list.  Then return.
;  524          0653  2      !
;  525          0654  2      FLINK = DBG$SCR_DISPLAY_LIST;
;  526          0655  2      BLINK = .FLINK[DBG$L_DISP_BLINK];
;  527          0656  2      DISPTR[DBG$L_DISP_FLINK] = .FLINK;
;  528          0657  2      DISPTR[DBG$L_DISP_BLINK] = .BLINK;
;  529          0658  2      FLINK[DBG$L_DISP_BLINK] = .DISPTR;
;  530          0659  2      BLINK[DBG$L_DISP_FLINK] = .DISPTR;
;  531          0660  2      RETURN .DISPTR;
;  532          0661  2
;  533          0662  1      END;


                                        007C 00000 DBG$SCR_CREATE_DISPLAY:
                                                   .WORD   Save R2,R3,R4,R5,R6                          ;  0565
                                04      AC  DD 00002       PUSHL   NAMEPTR                              ;  0621
                    0000V   CF          01  FB 00005       CALLS   #1, DBG$SCR_LOOKUP_DISPLAY
                            56          50  D0 0000A       MOVL    R0, DISPTR
                                        12  13 0000D       BEQL    1$                                   ;  0622
                                04      AC  DD 0000F       PUSHL   NAMEPTR
                                        01  DD 00012       PUSHL   #1
                                00028372 8F DD 00014       PUSHL   #164722
                    00000000G   00      03  FB 0001A       CALLS   #3, LIB$SIGNAL
                            50          04  BC  9A 00021 1$:    MOVZBL  @NAMEPTR, R0                    ;  0628
```

```
                        50              04  C0 00025          ADDL2   #4, R0                                  ;
                        50              04  C6 00028          DIVL2   #4, R0                                  ;
                                    13  A0  9F 0002B          PUSHAB  19(R0)                                  ;
                00000000G   00          01  FB 0002E          CALLS   #1, DBG$GET_MEMORY                      ;
                        56              50  D0 00035          MOVL    R0, DISPTR                              ;
                    08  A6          08  AC  90 00038          MOVB    KIND, 8(DISPTR)                         ; 0629
                        50          0C  AC  D0 0003D          MOVL    WPTR, R0                                ; 0630
                    10  A6          08  A0  7D 00041          MOVQ    8(R0), 16(DISPTR)                       ;
                    18  A6  00010001    8F  D0 00046          MOVL    #65537, 24(DISPTR)                      ; 0634
                        50          10  AC  D0 0004E          MOVL    SIZE, R0                                ; 0636
                                        03  14 00052          BGTR    2$                                      ;
                        50              01  D0 00054          MOVL    #1, R0                                  ;
                    1C  A6              50  B0 00057 2$:      MOVW    R0, 28(DISPTR)                          ;
                    0A  A6              22  88 0005B          BISB2   #34, 10(DISPTR)                         ; 0638
                    3C  A6              01  CE 0005F          MNEGL   #1, 60(DISPTR)                          ; 0639
                    48  A6          14  AC  D0 00063          MOVL    CONTENT_PTR, 72(DISPTR)                 ; 0640
                        50          04  BC  9A 00068          MOVZBL  @NAMEPTR, R0                            ; 0641
                                        50  D6 0006C          INCL    R0                                      ;
            4C  A6          04  BC          50  28 0006E      MOVC3   R0, @NAMEPTR, 76(DISPTR)                ;
                    20  A6          20  A6  9E 00074          MOVAB   32(DISPTR), 32(DISPTR)                  ; 0647
                    24  A6          20  A6  9E 00079          MOVAB   32(DISPTR), 36(DISPTR)                  ; 0648
                        50  00000000'   EF  9E 0007E          MOVAB   DBG$SCR_DISPLAY_LIST, FLINK             ; 0654
                        51          04  A0  D0 00085          MOVL    4(FLINK), BLINK                         ; 0655
                        66              50  7D 00089          MOVQ    FLINK, (DISPTR)                         ; 0656
                    04  A0              56  D0 0008C          MOVL    DISPTR, 4(FLINK)                        ; 0658
                        61              56  D0 00090          MOVL    DISPTR, (BLINK)                         ; 0659
                        50              56  D0 00093          MOVL    DISPTR, R0                              ; 0660
                                        04  00096            RET                                             ; 0662

; Routine Size:  151 bytes,    Routine Base:  DBG$CODE + 0A76
```

```
;  535     0663  1 ROUTINE DBG$SCR_CREATE_TEMP_WINDOW(RBEG, RLEN, CBEG, CLEN) =
;  536     0664  1 !
;  537     0665  1 ! FUNCTION
;  538     0666  1 !       This routine creates a Temporary Screen Window Entry.  A "temporary"
;  539     0667  1 !       Screen Window Entry is a Screen Window Entry created in temporary
;  540     0668  1 !       memory in which the window name is null.  Such entries are used for
;  541     0669  1 !       all unnamed windows used by DEBUG or specified by the user, that is
;  542     0670  1 !       for all window which are not named and put on the Screen Window List.
;  543     0671  1 !       All Temporary Screen Window Entries disappear at the end of the pro-
;  544     0672  1 !       cessing of the current command.
;  545     0673  1 !
;  546     0674  1 ! INPUTS
;  547     0675  1 !       RBEG    - The beginning row location on the screen of this window.
;  548     0676  1 !                 This value must be 1 or larger.
;  549     0677  1 !
;  550     0678  1 !       RLEN    - The row length (height) in lines of this window.
;  551     0679  1 !                 This value must be 1 or larger.
;  552     0680  1 !
;  553     0681  1 !       CBEG    - The beginning column location on the screen of this window.
;  554     0682  1 !                 This value must be 1 or larger.
;  555     0683  1 !
;  556     0684  1 !       CLEN    - The column length (width) in print positions of this window.
;  557     0685  1 !                 This value must be 1 or larger.
;  558     0686  1 !
;  559     0687  1 ! OUTPUTS
;  560     0688  1 !       A pointer to the new Temporary Screen Window Entry is returned as the
;  561     0689  1 !                 routine's value.
;  562     0690  1 !
;  563     0691  1
;  564     0692  2     BEGIN
;  565     0693  2
;  566     0694  2     LOCAL
;  567     0695  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to temporary Window Entry
;  568     0696  2
;  569     0697  2
;  570     0698  2
;  571     0699  2     ! Check that the window parameters have valid values.
;  572     0700  2     !
;  573     0701  2     IF (.RBEG LSS 1) OR (.RLEN LSS 1) OR
;  574     0702  3        (.CBEG LSS 1) OR (.CLEN LSS 1)
;  575     0703  2     THEN
;  576     0704  2         $DBG_ERROR('DBGSCREEN\CREATE_TEMP_WINDOW');
;  577     0705  2
;  578     0706  2
;  579     0707  2     ! Create a Temporary Screen Window Entry and fill in its contents.  Then
;  580     0708  2     ! return the address of the new Screen Window Entry to the caller.
;  581     0709  2     !
;  582     0710  2     WPTR = DBG$GET_TEMPMEM(DBG$K_WINDOW_ENTSIZE + 1);
;  583     0711  2     WPTR[DBG$W_WINDOW_RBEG] = .RBEG;
;  584     0712  2     WPTR[DBG$W_WINDOW_RLEN] = .RLEN;
;  585     0713  2     WPTR[DBG$W_WINDOW_CBEG] = .CBEG;
;  586     0714  2     WPTR[DBG$W_WINDOW_CLEN] = .CLEN;
;  587     0715  2     RETURN .WPTR;
;  588     0716  2
;  589     0717  1     END;
```

```
                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

41 45 52 43 5C 4E 45 45 52 43 53 47 42 44 1C  003CA P.AEZ:  .ASCII  <28>\DBGSCREEN\<92>\CREATE_TEMP_WINDOW\      ;
   57 4F 44 4E 49 57 5F 50 4D 45 54 5F 45 54  003D9                                                            ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                 0000 00000 DBG$SCR_CREATE_TEMP_WINDOW:
                                                        .WORD   Save nothing                                    ; 0663
                                          04    AC   D5 00002          TSTL    RBEG                             ; 0701
                                                0F   15 00005          BLEQ    1$                               ;
                                          08    AC   D5 00007          TSTL    RLEN                             ;
                                                0A   15 0000A          BLEQ    1$                               ;
                                          0C    AC   D5 0000C          TSTL    CBEG                             ; 0702
                                                05   15 0000F          BLEQ    1$                               ;
                                          10    AC   D5 00011          TSTL    CLEN                             ;
                                                15   14 00014          BGTR    2$                               ;
                                00000000'       EF   9F 00016 1$:      PUSHAB  P.AEZ                            ; 0704
                                                01   DD 0001C          PUSHL   #1                               ;
                                00028362        8F   DD 0001E          PUSHL   #164706                          ;
                    00000000G   00              03   FB 00024          CALLS   #3, LIB$SIGNAL                   ;
                                                05   DD 0002B 2$:      PUSHL   #5                               ; 0710
                    00000000G   00              01   FB 0002D          CALLS   #1, DBG$GET_TEMPMEM              ;
                          08    A0        04    AC   B0 00034          MOVW    RBEG, 8(WPTR)                    ; 0711
                          0A    A0        08    AC   B0 00039          MOVW    RLEN, 10(WPTR)                   ; 0712
                          0C    A0        0C    AC   B0 0003E          MOVW    CBEG, 12(WPTR)                   ; 0713
                          0E    A0        10    AC   B0 00043          MOVW    CLEN, 14(WPTR)                   ; 0714
                                                04   00048          RET                                      ; 0717

; Routine Size:  73 bytes,    Routine Base:  DBG$CODE + 0B0D
```

```
;  591          0718  1 ROUTINE DBG$SCR_CREATE_WINDOW(NAMEPTR, RBEG, RLEN, CBEG, CLEN) =
;  592          0719  1 !
;  593          0720  1 ! FUNCTION
;  594          0721  1 !       This routine creates a Screen Window Entry.  It accepts a new window
;  595          0722  1 !       name and the window parameters for a new screen window as input,
;  596          0723  1 !       creates a Screen Window Entry for such a window, and adds the Window
;  597          0724  1 !       Entry to the Screen Window List.  If a previous definition for a window
;  598          0725  1 !       by the same name already exists on the Screen Window List, that previ-
;  599          0726  1 !       ous entry is deleted before the new entry is added to the list.
;  600          0727  1 !
;  601          0728  1 ! INPUTS
;  602          0729  1 !       NAMEPTR - A pointer to the Counted ASCII name of the screen window
;  603          0730  1 !                 to be created.
;  604          0731  1 !
;  605          0732  1 !       RBEG    - The beginning row location on the screen of this window.
;  606          0733  1 !                 This value must be 1 or larger.
;  607          0734  1 !
;  608          0735  1 !       RLEN    - The row length (height) in lines of this window.
;  609          0736  1 !                 This value must be 1 or larger.
;  610          0737  1 !
;  611          0738  1 !       CBEG    - The beginning column location on the screen of this window.
;  612          0739  1 !                 This value must be 1 or larger.
;  613          0740  1 !
;  614          0741  1 !       CLEN    - The column length (width) in print positions of this window.
;  615          0742  1 !                 This value must be 1 or larger.
;  616          0743  1 !
;  617          0744  1 ! OUTPUTS
;  618          0745  1 !       A pointer to the created Screen Window Entry is returned as the
;  619          0746  1 !       routine's value.
;  620          0747  1 !
;  621          0748  1
;  622          0749  2     BEGIN
;  623          0750  2
;  624          0751  2     MAP
;  625          0752  2         NAMEPTR: REF VECTOR[,BYTE];     ! Pointer to ASCIC name for window
;  626          0753  2
;  627          0754  2     LOCAL
;  628          0755  2         BLINK: REF DBG$WINDOW_ENTRY,    ! Pointer to the last Window Entry in
;  629          0756  2                                         !      the Screen Window List
;  630          0757  2         FLINK: REF DBG$WINDOW_ENTRY,    ! Pointer to the list head for the
;  631          0758  2                                         !      Screen Window List
;  632          0759  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to new Screen Window Entry
;  633          0760  2
;  634          0761  2
;  635          0762  2
;  636          0763  2     ! Check that the window parameters have valid values.
;  637          0764  2     !
;  638          0765  2     IF (.RBEG LSS 1) OR (.RLEN LSS 1) OR
;  639          0766  3        (.CBEG LSS 1) OR (.CLEN LSS 1)
;  640          0767  2     THEN
;  641          0768  2         $DBG_ERROR('DBGSCREEN\CREATE_WINDOW');
;  642          0769  2
;  643          0770  2
;  644          0771  2     ! Create a new Screen Window Entry and fill in its contents.
;  645          0772  2     !
;  646          0773  2     WPTR = DBG$GET_MEMORY(DBG$K_WINDOW_ENTSIZE + (.NAMEPTR[0] + %UPVAL)/%UPVAL);
;  647          0774  2     CH$MOVE(.NAMEPTR[0] + 1, NAMEPTR[0], WPTR[DBG$A_WINDOW_NAME]);
```

```
;  648          0775  2     WPTR[DBG$W_WINDOW_RBEG] = .RBEG;
;  649          0776  2     WPTR[DBG$W_WINDOW_RLEN] = .RLEN;
;  650          0777  2     WPTR[DBG$W_WINDOW_CBEG] = .CBEG;
;  651          0778  2     WPTR[DBG$W_WINDOW_CLEN] = .CLEN;
;  652          0779  2
;  653          0780  2
;  654          0781  2     ! Now delete the current screen window by the same name, if one exists.
;  655          0782  2     !
;  656          0783  2     DBG$SCR_DELETE_WINDOW(.NAMEPTR);
;  657          0784  2
;  658          0785  2
;  659          0786  2     ! Finally link the new Window Entry into the Screen Window List.  The new
;  660          0787  2     ! entry is added to the end of the Screen Window List.  Then return a
;  661          0788  2     ! pointer to the new Window Entry.
;  662          0789  2     !
;  663          0790  2     FLINK = DBG$SCR_WINDOW_LIST;
;  664          0791  2     BLINK = .FLINK[DBG$L_WINDOW_BLINK];
;  665          0792  2     WPTR[DBG$L_WINDOW_FLINK] = .FLINK;
;  666          0793  2     WPTR[DBG$L_WINDOW_BLINK] = .BLINK;
;  667          0794  2     FLINK[DBG$L_WINDOW_BLINK] = .WPTR;
;  668          0795  2     BLINK[DBG$L_WINDOW_FLINK] = .WPTR;
;  669          0796  2     RETURN .WPTR;
;  670          0797  2
;  671          0798  1     END;


                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

41 45 52 43 5C 4E 45 45 52 43 53 47 42 44 17  003E7 P.AFA:  .ASCII  <23>\DBGSCREEN\<92>\CREATE_WINDOW\                ;
                  57 4F 44 4E 49 57 5F 45 54  003F6                                                                 ;


                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                         007C 00000 DBG$SCR_CREATE_WINDOW:
                                                .WORD   Save R2,R3,R4,R5,R6                                       ; 0718
                                     08  AC  D5 00002          TSTL    RBEG                                         ; 0765
                                     0F  15  00005              BLEQ    1$                                           ;
                                     0C  AC  D5 00007          TSTL    RLEN                                         ;
                                     0A  15  0000A              BLEQ    1$                                           ;
                                     10  AC  D5 0000C          TSTL    CBEG                                         ; 0766
                                     05  15  0000F              BLEQ    1$                                           ;
                                     14  AC  D5 00011          TSTL    CLEN                                         ;
                                     15  14  00014              BGTR    2$                                           ;
                           00000000'     EF  9F 00016 1$:      PUSHAB  P.AFA                                        ; 0768
                                     01  DD 0001C              PUSHL   #1                                           ;
                           00028362      8F  DD 0001E          PUSHL   #164706                                      ;
                00000000G  00            03  FB 00024          CALLS   #3, LIB$SIGNAL                               ;
                           50        04  BC  9A 0002B 2$:      MOVZBL  @NAMEPTR, R0                                 ; 0773
                           50            04  C0 0002F          ADDL2   #4, R0                                       ;
                           50            04  C6 00032          DIVL2   #4, R0                                       ;
                                     04  A0  9F 00035          PUSHAB  4(R0)                                        ;
                00000000G  00            01  FB 00038          CALLS   #1, DBG$GET_MEMORY                           ;
                           56            50  D0 0003F          MOVL    R0, WPTR                                     ;
                           50        04  BC  9A 00042          MOVZBL  @NAMEPTR, R0                                 ; 0774
```

```
                                                 50  D6 00046          INCL    R0                                              ;
                10   A6        04   BC           50  28 00048          MOVC3   R0, @NAMEPTR, 16(WPTR)                          ;
                               08   A6       08   AC  B0 0004E          MOVW    RBEG, 8(WPTR)                                   ; 0775
                               0A   A6       0C   AC  B0 00053          MOVW    RLEN, 10(WPTR)                                  ; 0776
                               0C   A6       10   AC  B0 00058          MOVW    CBEG, 12(WPTR)                                  ; 0777
                               0E   A6       14   AC  B0 0005D          MOVW    CLEN, 14(WPTR)                                  ; 0778
                                            04   AC  DD 00062          PUSHL   NAMEPTR                                         ; 0783
                          0000V CF                  01  FB 00065          CALLS   #1, DBG$SCR_DELETE_WINDOW                       ;
                                50 00000000'        EF  9E 0006A          MOVAB   DBG$SCR_WINDOW_LIST, FLINK                      ; 0790
                                51            04    A0  D0 00071          MOVL    4(FLINK), BLINK                                 ; 0791
                                66                  50  7D 00075          MOVQ    FLINK, (WPTR)                                   ; 0792
                             04 A0                  56  D0 00078          MOVL    WPTR, 4(FLINK)                                  ; 0794
                                61                  56  D0 0007C          MOVL    WPTR, (BLINK)                                   ; 0795
                                50                  56  D0 0007F          MOVL    WPTR, R0                                        ; 0796
                                                    04  00082          RET                                                     ; 0798

; Routine Size:  131 bytes,    Routine Base:  DBG$CODE + 0B56
```

```
;  673     0799  1 ROUTINE DBG$SCR_DELETE_DISPLAY(NAMEPTR) =
;  674     0800  1 !
;  675     0801  1 ! FUNCTION
;  676     0802  1 !       This routine deletes a specified screen display.  It accepts the name
;  677     0803  1 !       of the display to be deleted as input, looks up the corresponding
;  678     0804  1 !       Screen Display Entry, and removes that entry from the Screen Display
;  679     0805  1 !       List.  It then releases the Display Entry and all associated memory
;  680     0806  1 !       blocks back to the DEBUG memory pool.
;  681     0807  1 !
;  682     0808  1 ! INPUTS
;  683     0809  1 !       NAMEPTR - A pointer to the name of the display to be deleted.  The
;  684     0810  1 !                 name is represented as a Counted ASCII string.
;  685     0811  1 !
;  686     0812  1 ! OUTPUTS
;  687     0813  1 !       One of the following two status codes is returned as the routine value:
;  688     0814  1 !
;  689     0815  1 !               STS$K_SUCCESS   - The specified display was deleted.
;  690     0816  1 !
;  691     0817  1 !               STS$K_SEVERE    - No display by the specified name exists and
;  692     0818  1 !                                 hence no display was deleted.
;  693     0819  1 !
;  694     0820  1
;  695     0821  2     BEGIN
;  696     0822  2
;  697     0823  2     LOCAL
;  698     0824  2         BLINK: REF DBG$DISP_ENTRY,      ! Pointer to the previous Display Entry
;  699     0825  2                                         !      on the Screen Display List
;  700     0826  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to Display Entry to delete
;  701     0827  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to current Display Line Entry
;  702     0828  2         FLINK: REF DBG$DISP_ENTRY,      ! Pointer to the next Display Entry on
;  703     0829  2                                         !      the Screen Display List
;  704     0830  2         NEXTDLE: REF DBG$DLINE_ENTRY;   ! Pointer to next Display Line Entry
;  705     0831  2
;  706     0832  2
;  707     0833  2
;  708     0834  2     ! Look up the Screen Display Entry to be deleted.  If no display by the
;  709     0835  2     ! desired name exists, return unsuccessfully.
;  710     0836  2     !
;  711     0837  2     DISPTR = DBG$SCR_LOOKUP_DISPLAY(.NAMEPTR);
;  712     0838  2     IF .DISPTR EQL 0 THEN RETURN STS$K_SEVERE;
;  713     0839  2
;  714     0840  2
;  715     0841  2     ! We found the desired display.  If this display is the selected current
;  716     0842  2     ! input, output, scrolling, or source display, remove that selection by
;  717     0843  2     ! zeroing the associated pointer to the Display Entry.
;  718     0844  2     !
;  719     0845  2     IF .DISPTR EQL .DBG$SCR_CURDISP_INPUT  THEN DBG$SCR_CURDISP_INPUT  = 0;
;  720     0846  2     IF .DISPTR EQL .DBG$SCR_CURDISP_OUTPUT THEN DBG$SCR_CURDISP_OUTPUT = 0;
;  721     0847  2     IF .DISPTR EQL .DBG$SCR_CURDISP_SCROLL THEN DBG$SCR_CURDISP_SCROLL = 0;
;  722     0848  2     IF .DISPTR EQL .DBG$SCR_CURDISP_SOURCE THEN DBG$SCR_CURDISP_SOURCE = 0;
;  723     0849  2     IF .DISPTR EQL .DBG$GL_SCREEN_ERROR    THEN DBG$GL_SCREEN_ERROR    = 0;
;  724     0850  2     IF .DISPTR EQL .DBG$GL_SCREEN_HISTORY THEN DBG$GL_SCREEN_HISTORY = 0;
;  725     0851  2     IF .DISPTR EQL .DBG$GL_SCREEN_INPUT    THEN DBG$GL_SCREEN_INPUT    = 0;
;  726     0852  2     IF .DISPTR EQL .DBG$GL_SCREEN_OUTPUT  THEN DBG$GL_SCREEN_OUTPUT  = 0;
;  727     0853  2     IF .DISPTR EQL .DBG$GL_SCREEN_SOURCE  THEN DBG$GL_SCREEN_SOURCE  = 0;
;  728     0854  2
;  729     0855  2
```

```
;  730          0856  2      ! Now unlink the Display Entry from the Screen Display List.
;  731          0857  2      !
;  732          0858  2      FLINK = .DISPTR[DBG$L_DISP_FLINK];
;  733          0859  2      BLINK = .DISPTR[DBG$L_DISP_BLINK];
;  734          0860  2      FLINK[DBG$L_DISP_BLINK] = .BLINK;
;  735          0861  2      BLINK[DBG$L_DISP_FLINK] = .FLINK;
;  736          0862  2
;  737          0863  2
;  738          0864  2      ! Release the DEBUG command list entry for this display (if any) back to
;  739          0865  2      ! the DEBUG memory pool.
;  740          0866  2      !
;  741          0867  2      IF .DISPTR[DBG$L_DISP_CMDLIST] NEQ 0
;  742          0868  2      THEN
;  743          0869  2          DBG$REL_MEMORY(.DISPTR[DBG$L_DISP_CMDLIST]);
;  744          0870  2
;  745          0871  2
;  746          0872  2      ! Release all the Screen Display Line Entries (all the actual text lines)
;  747          0873  2      ! that belong to this screen display.
;  748          0874  2      !
;  749          0875  2      DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  750          0876  2      WHILE .DLEPTR NEQ DISPTR[DBG$L_DISP_START_LINE_PTR] DO
;  751          0877  3          BEGIN
;  752          0878  3          NEXTDLE = .DLEPTR[DBG$L_DLINE_FLINK];
;  753          0879  3          DBG$REL_MEMORY(.DLEPTR);
;  754          0880  3          DLEPTR = .NEXTDLE;
;  755          0881  2          END;
;  756          0882  2
;  757          0883  2
;  758          0884  2      ! Release all error Display Line Entries (if any) attached to this Screen
;  759          0885  2      ! Display Entry.
;  760          0886  2      !
;  761          0887  2      DLEPTR = .DISPTR[DBG$L_DISP_ERROR_PTR];
;  762          0888  2      WHILE .DLEPTR NEQ 0 DO
;  763          0889  3          BEGIN
;  764          0890  3          NEXTDLE = .DLEPTR[DBG$L_DLINE_FLINK];
;  765          0891  3          DBG$REL_MEMORY(.DLEPTR);
;  766          0892  3          DLEPTR = .NEXTDLE;
;  767          0893  2          END;
;  768          0894  2
;  769          0895  2
;  770          0896  2      ! Release all old-text Display Line Entries attached to this Display Entry.
;  771          0897  2      !
;  772          0898  2      DLEPTR = .DISPTR[DBG$L_DISP_OLDTXT_PTR];
;  773          0899  2      WHILE .DLEPTR NEQ 0 DO
;  774          0900  3          BEGIN
;  775          0901  3          NEXTDLE = .DLEPTR[DBG$L_DLINE_FLINK];
;  776          0902  3          DBG$REL_MEMORY(.DLEPTR);
;  777          0903  3          DLEPTR = .NEXTDLE;
;  778          0904  2          END;
;  779          0905  2
;  780          0906  2
;  781          0907  2      ! Finally release the Screen Display Entry itself to the memory pool.  Then
;  782          0908  2      ! return successfully to the caller.
;  783          0909  2      !
;  784          0910  2      DBG$REL_MEMORY(.DISPTR);
;  785          0911  2      RETURN STS$K_SUCCESS;
;  786          0912  2
```

```
;  787          0913  1     END;


                                00FC 00000 DBG$SCR_DELETE_DISPLAY:
                                                .WORD   Save R2,R3,R4,R5,R6,R7                  ; 0799
                            57 00000000G   00   9E 00002        MOVAB   DBG$REL_MEMORY, R7
                            56 00000000'   EF   9E 00009        MOVAB   DBG$SCR_CURDISP_INPUT, R6
                            55 00000000'   EF   9E 00010        MOVAB   DBG$GL_SCREEN_ERROR, R5
                                     04    AC   DD 00017        PUSHL   NAMEPTR                 ; 0837
                      0000V CF              01   FB 0001A        CALLS   #1, DBG$SCR_LOOKUP_DISPLAY
                            52              50   D0 0001F        MOVL    R0, DISPTR
                                            04   12 00022        BNEQ    1$                 ; 0838
                            50              04   D0 00024        MOVL    #4, R0
                                            04   00027        RET
                            66              52   D1 00028 1$:    CMPL    DISPTR, DBG$SCR_CURDISP_INPUT  ; 0845
                                            02   12 0002B        BNEQ    2$
                                            66   D4 0002D        CLRL    DBG$SCR_CURDISP_INPUT
                         04 A6              52   D1 0002F 2$:    CMPL    DISPTR, DBG$SCR_CURDISP_OUTPUT  ; 0846
                                            03   12 00033        BNEQ    3$
                                     04 A6  D4   00035        CLRL    DBG$SCR_CURDISP_OUTPUT
                         08 A6              52   D1 00038 3$:    CMPL    DISPTR, DBG$SCR_CURDISP_SCROLL  ; 0847
                                            03   12 0003C        BNEQ    4$
                                     08 A6  D4   0003E        CLRL    DBG$SCR_CURDISP_SCROLL
                         0C A6              52   D1 00041 4$:    CMPL    DISPTR, DBG$SCR_CURDISP_SOURCE  ; 0848
                                            03   12 00045        BNEQ    5$
                                     0C A6  D4   00047        CLRL    DBG$SCR_CURDISP_SOURCE
                            65              52   D1 0004A 5$:    CMPL    DISPTR, DBG$GL_SCREEN_ERROR  ; 0849
                                            02   12 0004D        BNEQ    6$
                                            65   D4 0004F        CLRL    DBG$GL_SCREEN_ERROR
                      04 A5                 52   D1 00051 6$:    CMPL    DISPTR, DBG$GL_SCREEN_HISTORY  ; 0850
                                            03   12 00055        BNEQ    7$
                                     04 A5  D4   00057        CLRL    DBG$GL_SCREEN_HISTORY
                      08 A5                 52   D1 0005A 7$:    CMPL    DISPTR, DBG$GL_SCREEN_INPUT  ; 0851
                                            03   12 0005E        BNEQ    8$
                                     08 A5  D4   00060        CLRL    DBG$GL_SCREEN_INPUT
                      18 A5                 52   D1 00063 8$:    CMPL    DISPTR, DBG$GL_SCREEN_OUTPUT  ; 0852
                                            03   12 00067        BNEQ    9$
                                     18 A5  D4   00069        CLRL    DBG$GL_SCREEN_OUTPUT
                      1C A5                 52   D1 0006C 9$:    CMPL    DISPTR, DBG$GL_SCREEN_SOURCE  ; 0853
                                            03   12 00070        BNEQ    10$
                                     1C A5  D4   00072        CLRL    DBG$GL_SCREEN_SOURCE
                            50              62   7D 00075 10$:   MOVQ    (DISPTR), FLINK         ; 0858
                      04    A0              51   D0 00078        MOVL    BLINK, 4(FLINK)         ; 0860
                            61              50   D0 0007C        MOVL    FLINK, (BLINK)          ; 0861
                                     48 A2  D5   0007F        TSTL    72(DISPTR)              ; 0867
                                            06   13 00082        BEQL    11$
                                     48 A2  DD   00084        PUSHL   72(DISPTR)              ; 0869
                            67              01   FB 00087        CALLS   #1, DBG$REL_MEMORY
                            53        20 A2 D0   0008A 11$:    MOVL    32(DISPTR), DLEPTR      ; 0875
                            50        20 A2 9E   0008E 12$:    MOVAB   32(DISPTR), R0          ; 0876
                            50              53   D1 00092        CMPL    DLEPTR, R0
                                            0D   13 00095        BEQL    13$
                            54              63   D0 00097        MOVL    (DLEPTR), NEXTDLE       ; 0878
                                            53   DD 0009A        PUSHL   DLEPTR                  ; 0879
```

```
                67              01  FB 0009C          CALLS   #1, DBG$REL_MEMORY                    ;
                53              54  D0 0009F          MOVL    NEXTDLE, DLEPTR                      ; 0880
                                EA  11 000A2          BRB     12$                                  ; 0876
                53      2C      A2  D0 000A4 13$:     MOVL    44(DISPTR), DLEPTR                   ; 0887
                                0D  13 000A8 14$:     BEQL    15$                                  ; 0888
                54              63  D0 000AA          MOVL    (DLEPTR), NEXTDLE                    ; 0890
                                53  DD 000AD          PUSHL   DLEPTR                               ; 0891
                67              01  FB 000AF          CALLS   #1, DBG$REL_MEMORY                   ;
                53              54  D0 000B2          MOVL    NEXTDLE, DLEPTR                      ; 0892
                                F1  11 000B5          BRB     14$                                  ; 0888
                53      30      A2  D0 000B7 15$:     MOVL    48(DISPTR), DLEPTR                   ; 0898
                                0D  13 000BB 16$:     BEQL    17$                                  ; 0899
                54              63  D0 000BD          MOVL    (DLEPTR), NEXTDLE                    ; 0901
                                53  DD 000C0          PUSHL   DLEPTR                               ; 0902
                67              01  FB 000C2          CALLS   #1, DBG$REL_MEMORY                   ;
                53              54  D0 000C5          MOVL    NEXTDLE, DLEPTR                      ; 0903
                                F1  11 000C8          BRB     16$                                  ; 0899
                                52  DD 000CA 17$:     PUSHL   DISPTR                               ; 0910
                67              01  FB 000CC          CALLS   #1, DBG$REL_MEMORY                   ;
                50              01  D0 000CF          MOVL    #1, R0                               ; 0911
                                    04 000D2          RET                                          ; 0913

; Routine Size:  211 bytes,    Routine Base:  DBG$CODE + 0BD9
```

```
;  789     0914  1 ROUTINE DBG$SCR_DELETE_WINDOW(NAMEPTR) =
;  790     0915  1 !
;  791     0916  1 ! FUNCTION
;  792     0917  1 !       This routine deletes a specified screen window definition.  It accepts
;  793     0918  1 !       the name of the window to be deleted as input, looks up the correspond-
;  794     0919  1 !       ing Screen Window Entry, and removes that entry from the Screen Window
;  795     0920  1 !       List.  It then releases the Window Entry back to the DEBUG memory pool.
;  796     0921  1 !
;  797     0922  1 ! INPUTS
;  798     0923  1 !       NAMEPTR - A pointer to the name of the window to be deleted.  The name
;  799     0924  1 !                 is represented as a Counted ASCII string.
;  800     0925  1 !
;  801     0926  1 ! OUTPUTS
;  802     0927  1 !       One of the following two status codes is returned as the routine value:
;  803     0928  1 !
;  804     0929  1 !               STS$K_SUCCESS   - The specified window definition was deleted.
;  805     0930  1 !
;  806     0931  1 !               STS$K_SEVERE    - No window by the specified name exists and
;  807     0932  1 !                                 hence no window was deleted.
;  808     0933  1 !
;  809     0934  1
;  810     0935  2     BEGIN
;  811     0936  2
;  812     0937  2     LOCAL
;  813     0938  2         BLINK: REF DBG$WINDOW_ENTRY,    ! Pointer to the previous Window Entry
;  814     0939  2                                         !      on the Screen Window List
;  815     0940  2         FLINK: REF DBG$WINDOW_ENTRY,    ! Pointer to the next Window Entry on
;  816     0941  2                                         !      the Screen Window List
;  817     0942  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to the Window Entry to delete
;  818     0943  2
;  819     0944  2
;  820     0945  2
;  821     0946  2     ! Look up the Screen Window Entry to be deleted.  If no window by the
;  822     0947  2     ! desired name exists, return unsuccessfully.
;  823     0948  2     !
;  824     0949  2     WPTR = DBG$SCR_LOOKUP_WINDOW(.NAMEPTR);
;  825     0950  2     IF .WPTR EQL 0 THEN RETURN STS$K_SEVERE;
;  826     0951  2
;  827     0952  2
;  828     0953  2     ! We found the desired window definition.  Unlink that Window Entry from
;  829     0954  2     ! the Screen Window List.
;  830     0955  2     !
;  831     0956  2     FLINK = .WPTR[DBG$L_WINDOW_FLINK];
;  832     0957  2     BLINK = .WPTR[DBG$L_WINDOW_BLINK];
;  833     0958  2     FLINK[DBG$L_WINDOW_BLINK] = .BLINK;
;  834     0959  2     BLINK[DBG$L_WINDOW_FLINK] = .FLINK;
;  835     0960  2
;  836     0961  2
;  837     0962  2     ! Release the Screen Window Entry back to the memory pool.  Then return
;  838     0963  2     ! successfully to the caller.
;  839     0964  2     !
;  840     0965  2     DBG$REL_MEMORY(.WPTR);
;  841     0966  2     RETURN STS$K_SUCCESS;
;  842     0967  2
;  843     0968  1     END;
```

```
                                 0004 00000 DBG$SCR_DELETE_WINDOW:
                                            .WORD   Save R2                                 ; 0914
                          04     AC  DD 00002        PUSHL   NAMEPTR                        ; 0949
                    0000V CF         01  FB 00005    CALLS   #1, DBG$SCR_LOOKUP_WINDOW
                                     50  D5 0000A    TSTL    WPTR                           ; 0950
                                     04  12 0000C    BNEQ    1$
                          50         04  D0 0000E    MOVL    #4, R0
                                     04  00011      RET
                          51         60  7D 00012 1$: MOVQ    (WPTR), FLINK                  ; 0956
                       04 A1         52  D0 00015    MOVL    BLINK, 4(FLINK)                ; 0958
                          62         51  D0 00019    MOVL    FLINK, (BLINK)                 ; 0959
                                     50  DD 0001C    PUSHL   WPTR                           ; 0965
                00000000G 00         01  FB 0001E    CALLS   #1, DBG$REL_MEMORY
                          50         01  D0 00025    MOVL    #1, R0                         ; 0966
                                     04  00028      RET                                    ; 0968

; Routine Size:  41 bytes,    Routine Base:  DBG$CODE + 0CAC
```

```
;  845          0969  1 ROUTINE DBG$SCR_DISPLAY_COMMAND(NAMEPTR, KIND, WPTR, SIZE,
;  846          0970  1                                  CONTENT_PTR, LIST_LOCATION) =
;  847          0971  1 !
;  848          0972  1 ! FUNCTION
;  849          0973  1 !       This routine performs the semantic actions of the DISPLAY command.  It
;  850          0974  1 !       accepts the name of an existing display and its desired attributes as
;  851          0975  1 !       input.  It then modifies all attributes of the display as specified by
;  852          0976  1 !       the input parameters.  This may include changing the display's window,
;  853          0977  1 !       its kind, its size, and its location on the display list (which affects
;  854          0978  1 !       it "depth" on the screen relative to other displays).
;  855          0979  1 !
;  856          0980  1 ! INPUTS
;  857          0981  1 !       NAMEPTR - A pointer to the name of the display to be displayed.  The
;  858          0982  1 !                 name is represented as a Counted ASCII string.
;  859          0983  1 !
;  860          0984  1 !       KIND    - The new kind of the display (if the kind is being changed)
;  861          0985  1 !                 or DBG$K_DISP_NOKIND (if the kind is not being changed).
;  862          0986  1 !                 This basically refers to how the contents are generated:
;  863          0987  1 !                 it may be NORMAL, SOURCE, DO, etc.
;  864          0988  1 !
;  865          0989  1 !       WPTR    - A pointer to a Screen Window Entry if the display's window
;  866          0990  1 !                 parameters are being changed.  This Window Entry defines
;  867          0991  1 !                 the new screen window for the display.  If the window is
;  868          0992  1 !                 not being changed, WPTR is zero.
;  869          0993  1 !
;  870          0994  1 !       SIZE    - The new maximum size of the display in text lines (if the
;  871          0995  1 !                 display size is being changed) or zero (if the size is not
;  872          0996  1 !                 being changed).
;  873          0997  1 !
;  874          0998  1 !       CONTENT_PTR - A pointer to the DEBUG command list which determines
;  875          0999  1 !                 the contents of this display if this is an automatically
;  876          1000  1 !                 updated display.  If this is not an automatically updated
;  877          1001  1 !                 display according to the KIND parameter or if the KIND
;  878          1002  1 !                 parameter has the value DBG$K_DISP_NOKIND (no change to
;  879          1003  1 !                 the display kind), then CONTENT_PTR is ignored and should
;  880          1004  1 !                 be zero.
;  881          1005  1 !
;  882          1006  1 !       LIST_LOCATION - A parameter which specifies the display's location
;  883          1007  1 !                 on the Screen Display List.  If LIST_LOCATION is zero, the
;  884          1008  1 !                 list location is not changed.  If it is positive, the dis-
;  885          1009  1 !                 play is put at the front of the list (so it occludes other
;  886          1010  1 !                 displays on the screen), and if it is negative, the display
;  887          1011  1 !                 is put at the back of the list (so that it is occluded by
;  888          1012  1 !                 by any overlapping displays on the screen).
;  889          1013  1 !
;  890          1014  1 ! OUTPUTS
;  891          1015  1 !       A pointer to the Screen Display Entry of the specified display is
;  892          1016  1 !                 returned as this routine's value.
;  893          1017  1 !
;  894          1018  1
;  895          1019  2     BEGIN
;  896          1020  2
;  897          1021  2     MAP
;  898          1022  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to Screen Window Entry
;  899          1023  2
;  900          1024  2     LOCAL
;  901          1025  2         BLINK: REF DBG$DISP_ENTRY,      ! Pointer to previous Screen Display
```

```
;  902          1026  2                                    !       Entry on the Screen Display List
;  903          1027  2          DISPTR: REF DBG$DISP_ENTRY,    ! Pointer to new Screen Display Entry
;  904          1028  2          DLEPTR: REF DBG$DLINE_ENTRY,   ! Pointer to Display Line Entry to remove
;  905          1029  2          DL_BLINK: REF DBG$DLINE_ENTRY, ! Pointer to previous Display Line Entry
;  906          1030  2          DL_FLINK: REF DBG$DLINE_ENTRY, ! Pointer to next Display Line Entry
;  907          1031  2          FLINK: REF DBG$DISP_ENTRY,     ! Pointer to next Screen Display Entry
;  908          1032  2                                         !       on the Screen Display List
;  909          1033  2          NEW_RLEN,                      ! New RLEN value when changing window
;  910          1034  2          OLD_RLEN;                      ! Old RLEN value when changing window
;  911          1035  2
;  912          1036  2
;  913          1037  2
;  914          1038  2      ! Look up the desired display by its name.  If no such display exists,
;  915          1039  2      ! signal an error (no such screen display).
;  916          1040  2      !
;  917          1041  2      DISPTR = DBG$SCR_LOOKUP_DISPLAY(.NAMEPTR);
;  918          1042  2      IF .DISPTR EQL 0 THEN SIGNAL(DBG$_NOSUCHDISP, 1, .NAMEPTR);
;  919          1043  2
;  920          1044  2
;  921          1045  2      ! If the display kind is being changed, set the new values of the kind
;  922          1046  2      ! and command-list fields in the Screen Display Entry.  Note that we
;  923          1047  2      ! first clear the contents if we are switching from a source display to
;  924          1048  2      ! a non-source display or vice versa (normal and source Display Line
;  925          1049  2      ! Entries cannot be mixed in the same display).  We also release the old
;  926          1050  2      ! command-list block if one exists.
;  927          1051  2      !
;  928          1052  2      IF .KIND NEQ DBG$K_DISP_NOKIND
;  929          1053  2      THEN
;  930          1054  3          BEGIN
;  931          1055  3
;  932          1056  3
;  933          1057  3          ! If we are changing from a non-source display to a source display, make
;  934          1058  3          ! sure that this display is no longer the current input or output dis-
;  935          1059  3          ! play.  Also clear its contents so that no normal Display Line Entries
;  936          1060  3          ! are left in a source display.
;  937          1061  3          !
;  938          1062  3          IF (.DISPTR[DBG$B_DISP_KIND] NEQ DBG$K_DISP_SOURCE) AND
;  939          1063  4             (.KIND EQL DBG$K_DISP_SOURCE)
;  940          1064  3          THEN
;  941          1065  4              BEGIN
;  942          1066  4              IF .DISPTR EQL .DBG$SCR_CURDISP_OUTPUT THEN DBG$SCR_CURDISP_OUTPUT = 0;
;  943          1067  4              IF .DISPTR EQL .DBG$SCR_CURDISP_INPUT THEN DBG$SCR_CURDISP_INPUT = 0;
;  944          1068  4              IF .DISPTR EQL .DBG$GL_SCREEN_OUTPUT THEN DBG$GL_SCREEN_OUTPUT = 0;
;  945          1069  4              IF .DISPTR EQL .DBG$GL_SCREEN_INPUT THEN DBG$GL_SCREEN_INPUT = 0;
;  946          1070  4              DBG$SCR_EMPTY_DISPLAY(.DISPTR);
;  947          1071  3              END;
;  948          1072  3
;  949          1073  3
;  950          1074  3          ! If we are changing from a source display to a non-source display, make
;  951          1075  3          ! sure that this display is no longer the current source display.  Clear
;  952          1076  3          ! its contents so that no source Display Line Entries are left in a nor-
;  953          1077  3          ! mal output display.  Also zero out all source-display specific fields
;  954          1078  3          ! in the Screen Display Entry.
;  955          1079  3          !
;  956          1080  3          IF (.DISPTR[DBG$B_DISP_KIND] EQL DBG$K_DISP_SOURCE) AND
;  957          1081  4             (.KIND NEQ DBG$K_DISP_SOURCE)
;  958          1082  3          THEN
```

```
;  959          1083  4              BEGIN
;  960          1084  4              IF .DISPTR EQL .DBG$SCR_CURDISP_SOURCE THEN DBG$SCR_CURDISP_SOURCE = 0;
;  961          1085  4              IF .DISPTR EQL .DBG$GL_SCREEN_SOURCE THEN DBG$GL_SCREEN_SOURCE = 0;
;  962          1086  4              DBG$SCR_EMPTY_DISPLAY(.DISPTR);
;  963          1087  4              DISPTR[DBG$L_DISP_MODPTR] = 0;
;  964          1088  4              DISPTR[DBG$L_DISP_CENTER] = 0;
;  965          1089  4              DISPTR[DBG$L_DISP_MARKLINE] = 0;
;  966          1090  4              DISPTR[DBG$L_DISP_MINLINE] = 0;
;  967          1091  4              DISPTR[DBG$L_DISP_MAXLINE] = 0;
;  968          1092  3              END;
;  969          1093  3
;  970          1094  3
;  971          1095  3          ! Remove the old DEBUG command list, if present.  The change the Display
;  972          1096  3          ! Entry's kind field and set the new DEBUG command list address.
;  973          1097  3          !
;  974          1098  3          IF .DISPTR[DBG$L_DISP_CMDLIST] NEQ 0
;  975          1099  3          THEN
;  976          1100  3              DBG$REL_MEMORY(.DISPTR[DBG$L_DISP_CMDLIST]);
;  977          1101  3
;  978          1102  3          DISPTR[DBG$B_DISP_KIND] = .KIND;
;  979          1103  3          DISPTR[DBG$L_DISP_CMDLIST] = .CONTENT_PTR;
;  980          1104  2          END;
;  981          1105  2
;  982          1106  2
;  983          1107  2      ! If the display window parameters are being changed, pick up the new
;  984          1108  2      ! window parameter values from the specified Screen Window Entry.
;  985          1109  2      !
;  986          1110  2      IF .WPTR NEQ 0
;  987          1111  2      THEN
;  988          1112  3          BEGIN
;  989          1113  3
;  990          1114  3
;  991          1115  3          ! Set the new values of the window parameters for this display.
;  992          1116  3          !
;  993          1117  3          OLD_RLEN = .DISPTR[DBG$W_DISP_RLEN];
;  994          1118  3          NEW_RLEN = .WPTR[DBG$W_WINDOW_RLEN];
;  995          1119  3          DISPTR[DBG$W_DISP_RBEG] = .WPTR[DBG$W_WINDOW_RBEG];
;  996          1120  3          DISPTR[DBG$W_DISP_CBEG] = .WPTR[DBG$W_WINDOW_CBEG];
;  997          1121  3          DISPTR[DBG$W_DISP_RLEN] = .WPTR[DBG$W_WINDOW_RLEN];
;  998          1122  3          DISPTR[DBG$W_DISP_CLEN] = .WPTR[DBG$W_WINDOW_CLEN];
;  999          1123  3
; 1000          1124  3
; 1001          1125  3          ! Adjust the DROW value (window row location in the display text) and
; 1002          1126  3          ! the window start pointer to fit the new vertical size of the display.
; 1003          1127  3          ! Essentially we try to keep the window centered on the same text as
; 1004          1128  3          ! before.  We also invalidate the scrolling count since it is no longer
; 1005          1129  3          ! usable by the OUTPUT_SCREEN routine.
; 1006          1130  3          !
; 1007          1131  3          DISPTR[DBG$V_DISP_INVSCR] = TRUE;
; 1008          1132  3          DISPTR[DBG$W_DISP_DROW] = MAX(1, MIN(
; 1009          1133  3                  .DISPTR[DBG$W_DISP_LINECNT] - .NEW_RLEN + 1,
; 1010          1134  3                  .DISPTR[DBG$W_DISP_DROW] + (.OLD_RLEN - .NEW_RLEN)/2));
; 1011          1135  3          IF .DISPTR[DBG$W_DISP_LINECNT] GTR 0
; 1012          1136  3          THEN
; 1013          1137  4              BEGIN
; 1014          1138  4              DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
; 1015          1139  4              INCR I FROM 1 TO .DISPTR[DBG$W_DISP_DROW] - 1 DO
```

```
;  1016          1140  4                DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  1017          1141  4
;  1018          1142  4            DISPTR[DBG$L_DISP_WINDOW_PTR] = .DLEPTR;
;  1019          1143  3            END;
;  1020          1144  3
;  1021          1145  2        END;
;  1022          1146  2
;  1023          1147  2
;  1024          1148  2        ! If the display size (the maximum number of lines of text the display may
;  1025          1149  2        ! contain) is being changed, set the new size in the Screen Display Entry
;  1026          1150  2        ! and delete any existing lines which cause the line count to exceed the
;  1027          1151  2        ! new maximum.  Note that we always delete the oldest lines first.  Note
;  1028          1152  2        ! that the scrolling count is also invalidated.
;  1029          1153  2        !
;  1030          1154  2        IF .SIZE GTR 0
;  1031          1155  2        THEN
;  1032          1156  3            BEGIN
;  1033          1157  3            DISPTR[DBG$W_DISP_MAX_LINECNT] = .SIZE;
;  1034          1158  3            DISPTR[DBG$V_DISP_INVSCR] = TRUE;
;  1035          1159  3
;  1036          1160  3
;  1037          1161  3            ! Delete enough of the oldest lines to bring the display size down to
;  1038          1162  3            ! the new maximum size.
;  1039          1163  3            !
;  1040          1164  3            INCR I FROM .SIZE + 1 TO .DISPTR[DBG$W_DISP_LINECNT] DO
;  1041          1165  4                BEGIN
;  1042          1166  4
;  1043          1167  4
;  1044          1168  4                ! Delete the oldest line still in the display and decrement the
;  1045          1169  4                ! line count.
;  1046          1170  4                !
;  1047          1171  4                DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  1048          1172  4                DL_FLINK = .DLEPTR[DBG$L_DLINE_FLINK];
;  1049          1173  4                DL_BLINK = .DLEPTR[DBG$L_DLINE_BLINK];
;  1050          1174  4                DL_FLINK[DBG$L_DLINE_BLINK] = .DL_BLINK;
;  1051          1175  4                DL_BLINK[DBG$L_DLINE_FLINK] = .DL_FLINK;
;  1052          1176  4                DBG$REL_MEMORY(.DLEPTR);
;  1053          1177  4                DISPTR[DBG$W_DISP_LINECNT] = .DISPTR[DBG$W_DISP_LINECNT] - 1;
;  1054          1178  4
;  1055          1179  4
;  1056          1180  4                ! Decrement DROW so that the current window position is maintained.
;  1057          1181  4                ! However, if lines in the current window are being deleted, we
;  1058          1182  4                ! adjust DROW and WINDOW_PTR accordingly.
;  1059          1183  4                !
;  1060          1184  4                DISPTR[DBG$W_DISP_DROW] = .DISPTR[DBG$W_DISP_DROW] - 1;
;  1061          1185  4                IF .DISPTR[DBG$W_DISP_DROW] LEQ 0
;  1062          1186  4                THEN
;  1063          1187  5                    BEGIN
;  1064          1188  5                    DISPTR[DBG$W_DISP_DROW] = 1;
;  1065          1189  5                    DISPTR[DBG$L_DISP_WINDOW_PTR] = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  1066          1190  4                    END;
;  1067          1191  4
;  1068          1192  3            END;
;  1069          1193  3
;  1070          1194  2        END;
;  1071          1195  2
;  1072          1196  2
```

```
;  1073          1197  2     ! If the list location of this Display Entry is to be changed, unlink it
;  1074          1198  2     ! from the Screen Display List and then relink it into the proper place
;  1075          1199  2     ! (either at the beginning or at the end of the list).
;  1076          1200  2     !
;  1077          1201  2     IF .LIST_LOCATION NEQ 0
;  1078          1202  2     THEN
;  1079          1203  3         BEGIN
;  1080          1204  3
;  1081          1205  3
;  1082          1206  3         ! Unlink the specified Screen Display Entry from the Display List.
;  1083          1207  3         !
;  1084          1208  3         FLINK = .DISPTR[DBG$L_DISP_FLINK];
;  1085          1209  3         BLINK = .DISPTR[DBG$L_DISP_BLINK];
;  1086          1210  3         FLINK[DBG$L_DISP_BLINK] = .BLINK;
;  1087          1211  3         BLINK[DBG$L_DISP_FLINK] = .FLINK;
;  1088          1212  3
;  1089          1213  3
;  1090          1214  3         ! Determine the proper list location for reinserting the Display Entry.
;  1091          1215  3         !
;  1092          1216  3         BLINK = DBG$SCR_DISPLAY_LIST;
;  1093          1217  3         IF .LIST_LOCATION GTR 0 THEN BLINK = .BLINK[DBG$L_DISP_BLINK];
;  1094          1218  3         FLINK = .BLINK[DBG$L_DISP_FLINK];
;  1095          1219  3
;  1096          1220  3
;  1097          1221  3         ! Then reinsert the Display Entry into the Screen Display List.
;  1098          1222  3         !
;  1099          1223  3         DISPTR[DBG$L_DISP_FLINK] = .FLINK;
;  1100          1224  3         DISPTR[DBG$L_DISP_BLINK] = .BLINK;
;  1101          1225  3         FLINK[DBG$L_DISP_BLINK] = .DISPTR;
;  1102          1226  3         BLINK[DBG$L_DISP_FLINK] = .DISPTR;
;  1103          1227  2         END;
;  1104          1228  2
;  1105          1229  2
;  1106          1230  2     ! If the display kind changed so it now is automatically updated, we need
;  1107          1231  2     ! to regenerate its contents.  This is done by calling the screen display
;  1108          1232  2     ! generation routine.
;  1109          1233  2     !
;  1110          1234  2     IF (.KIND EQL DBG$K_DISP_DO)     OR
;  1111          1235  2        (.KIND EQL DBG$K_DISP_SOURCE) OR
;  1112          1236  3        (.KIND EQL DBG$K_DISP_REGISTER)
;  1113          1237  2     THEN
;  1114          1238  2         DBG$SCR_GENERATE_SCREEN(.DISPTR);
;  1115          1239  2
;  1116          1240  2
;  1117          1241  2     ! The DISPLAY command is fully processed.  Now return.
;  1118          1242  2     !
;  1119          1243  2     RETURN .DISPTR;
;  1120          1244  2
;  1121          1245  1     END;


                                OFFC 00000 DBG$SCR_DISPLAY_COMMAND:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 0969
                5B 00000000G   00   9E 00002      MOVAB   DBG$REL_MEMORY, R11                  ;
```

```
                5A 00000000'   EF   9E 00009          MOVAB   DBG$GL_SCREEN_OUTPUT, R10                    ;
                59 00000000'   EF   9E 00010          MOVAB   DBG$SCR_CURDISP_OUTPUT, R9                  ;
                        04     AC   DD 00017          PUSHL   NAMEPTR                                     ; 1041
          0000V CF             01   FB 0001A          CALLS   #1, DBG$SCR_LOOKUP_DISPLAY                  ;
                53             50   D0 0001F          MOVL    R0, DISPTR                                  ;
                               12   12 00022          BNEQ    1$                                          ; 1042
                        04     AC   DD 00024          PUSHL   NAMEPTR                                     ;
                               01   DD 00027          PUSHL   #1                                          ;
                00028A62       8F   DD 00029          PUSHL   #166498                                     ;
00000000G       00             03   FB 0002F          CALLS   #3, LIB$SIGNAL                              ;
                56             08   AC   D0 00036 1$:  MOVL    KIND, R6                                    ; 1052
                               73   13 0003A          BEQL    11$                                         ;
                03             08   A3   91 0003C     CMPB    8(DISPTR), #3                               ; 1062
                               2C   13 00040          BEQL    6$                                          ;
                03             56   D1 00042          CMPL    R6, #3                                      ; 1063
                               27   12 00045          BNEQ    6$                                          ;
                69             53   D1 00047          CMPL    DISPTR, DBG$SCR_CURDISP_OUTPUT              ; 1066
                               02   12 0004A          BNEQ    2$                                          ;
                               69   D4 0004C          CLRL    DBG$SCR_CURDISP_OUTPUT                      ;
              FC   A9          53   D1 0004E 2$:     CMPL    DISPTR, DBG$SCR_CURDISP_INPUT               ; 1067
                               03   12 00052          BNEQ    3$                                          ;
                        FC     A9   D4 00054          CLRL    DBG$SCR_CURDISP_INPUT                       ;
                6A             53   D1 00057 3$:     CMPL    DISPTR, DBG$GL_SCREEN_OUTPUT                ; 1068
                               02   12 0005A          BNEQ    4$                                          ;
                               6A   D4 0005C          CLRL    DBG$GL_SCREEN_OUTPUT                        ;
              F0   AA          53   D1 0005E 4$:     CMPL    DISPTR, DBG$GL_SCREEN_INPUT                 ; 1069
                               03   12 00062          BNEQ    5$                                          ;
                        F0     AA   D4 00064          CLRL    DBG$GL_SCREEN_INPUT                         ;
                               53   DD 00067 5$:     PUSHL   DISPTR                                      ; 1070
          0000V CF             01   FB 00069          CALLS   #1, DBG$SCR_EMPTY_DISPLAY                   ;
                03       08    A3   91 0006E 6$:     CMPB    8(DISPTR), #3                               ; 1080
                               27   12 00072          BNEQ    9$                                          ;
                03             56   D1 00074          CMPL    R6, #3                                      ; 1081
                               22   13 00077          BEQL    9$                                          ;
              08   A9          53   D1 00079          CMPL    DISPTR, DBG$SCR_CURDISP_SOURCE              ; 1084
                               03   12 0007D          BNEQ    7$                                          ;
                        08     A9   D4 0007F          CLRL    DBG$SCR_CURDISP_SOURCE                      ;
              04   AA          53   D1 00082 7$:     CMPL    DISPTR, DBG$GL_SCREEN_SOURCE                ; 1085
                               03   12 00086          BNEQ    8$                                          ;
                        04     AA   D4 00088          CLRL    DBG$GL_SCREEN_SOURCE                        ;
                               53   DD 0008B 8$:     PUSHL   DISPTR                                      ; 1086
          0000V CF             01   FB 0008D          CALLS   #1, DBG$SCR_EMPTY_DISPLAY                   ;
                        34     A3   7C 00092          CLRQ    52(DISPTR)                                  ; 1087
                        3C     A3   7C 00095          CLRQ    60(DISPTR)                                  ; 1089
                        44     A3   D4 00098          CLRL    68(DISPTR)                                  ; 1091
                        48     A3   D5 0009B 9$:      TSTL    72(DISPTR)                                  ; 1098
                               06   13 0009E          BEQL    10$                                         ;
                        48     A3   DD 000A0          PUSHL   72(DISPTR)                                  ; 1100
                68             01   FB 000A3          CALLS   #1, DBG$REL_MEMORY                          ;
              08   A3          56   90 000A6 10$:     MOVB    R6, 8(DISPTR)                               ; 1102
              48   A3    14    AC   D0 000AA          MOVL    CONTENT_PTR, 72(DISPTR)                     ; 1103
                50       0C    AC   D0 000AF 11$:     MOVL    WPTR, R0                                    ; 1110
                               56   13 000B3          BEQL    16$                                         ;
                51       12    A3   3C 000B5          MOVZWL  18(DISPTR), OLD_RLEN                        ; 1117
                52       0A    A0   3C 000B9          MOVZWL  10(R0), NEW_RLEN                            ; 1118
              10   A3    08    A0   7D 000BD          MOVQ    8(R0), 16(DISPTR)                           ; 1119
              0A   A3          02   88 000C2          BISB2   #2, 10(DISPTR)                              ; 1131
```

```
                50      1E  A3  3C 000C6          MOVZWL  30(DISPTR), R0                  ;1133
                50          52  C2 000CA          SUBL2   NEW_RLEN, R0                    ;
                            50  D6 000CD          INCL    R0                              ;
                51          52  C2 000CF          SUBL2   NEW_RLEN, R1                    ;1134
                51          02  C6 000D2          DIVL2   #2, R1                          ;
                54      18  A3  3C 000D5          MOVZWL  24(DISPTR), R4                  ;
                51          54  C0 000D9          ADDL2   R4, R1                          ;
                51          50  D1 000DC          CMPL    R0, R1                          ;
                            03  15 000DF          BLEQ    12$                             ;
                50          51  D0 000E1          MOVL    R1, R0                          ;
                            50  D5 000E4 12$:      TSTL    R0                              ;1132
                            03  14 000E6          BGTR    13$                             ;
                50          01  D0 000E8          MOVL    #1, R0                          ;
            18  A3          50  B0 000EB 13$:     MOVW    R0, 24(DISPTR)                  ;
                1E  A3      B5 000EF              TSTW    30(DISPTR)                      ;1135
                            17  13 000F2          BEQL    16$                             ;
                54      20  A3  D0 000F4          MOVL    32(DISPTR), DLEPTR              ;1138
                51      18  A3  3C 000F8          MOVZWL  24(DISPTR), R1                  ;1139
                            50  D4 000FC          CLRL    I                               ;
                            03  11 000FE          BRB     15$                             ;
                54          64  D0 00100 14$:     MOVL    (DLEPTR), DLEPTR                ;1140
        F9      50          51  F2 00103 15$:     AOBLSS  R1, I, 14$                      ;
            28  A3          54  D0 00107          MOVL    DLEPTR, 40(DISPTR)              ;1142
                52      10  AC  D0 0010B 16$:     MOVL    SIZE, R2                        ;1154
                            3A  15 0010F          BLEQ    19$                             ;
            1C  A3          52  B0 00111          MOVW    R2, 28(DISPTR)                  ;1157
            0A  A3          02  88 00115          BISB2   #2, 10(DISPTR)                  ;1158
                58      1E  A3  3C 00119          MOVZWL  30(DISPTR), R8                  ;1164
                            28  11 0011D          BRB     18$                             ;1184
                54      20  A3  D0 0011F 17$:     MOVL    32(DISPTR), DLEPTR              ;1171
                55          64  D0 00123          MOVL    (DLEPTR), DL_FLINK              ;1172
                57      04  A4  D0 00126          MOVL    4(DLEPTR), DL_BLINK             ;1173
            04  A5          57  D0 0012A          MOVL    DL_BLINK, 4(DL_FLINK)           ;1174
                67          55  D0 0012E          MOVL    DL_FLINK, (DL_BLINK)            ;1175
                            54  DD 00131          PUSHL   DLEPTR                          ;1176
                6B          01  FB 00133          CALLS   #1, DBG$REL_MEMORY              ;
                        1E  A3  B7 00136          DECW    30(DISPTR)                      ;1177
                        18  A3  B7 00139          DECW    24(DISPTR)                      ;1184
                            09  12 0013C          BNEQ    18$                             ;1185
            18  A3          01  B0 0013E          MOVW    #1, 24(DISPTR)                  ;1188
            28  A3      20  A3  D0 00142          MOVL    32(DISPTR), 40(DISPTR)          ;1189
        D4      52          58  F3 00147 18$:     AOBLEQ  R8, I, 17$                      ;1164
                        18  AC  D5 0014B 19$:     TSTL    LIST_LOCATION                   ;1201
                            24  13 0014E          BEQL    21$                             ;
                50          63  7D 00150          MOVQ    (DISPTR), FLINK                 ;1208
            04  A0          51  D0 00153          MOVL    BLINK, 4(FLINK)                 ;1210
                61          50  D0 00157          MOVL    FLINK, (BLINK)                  ;1211
                51      0C  A9  9E 0015A          MOVAB   DBG$SCR_DISPLAY_LIST, BLINK     ;1216
                        18  AC  D5 0015E          TSTL    LIST_LOCATION                   ;1217
                            04  15 00161          BLEQ    20$                             ;
                51      04  A1  D0 00163          MOVL    4(BLINK), BLINK                 ;
                50          61  D0 00167 20$:     MOVL    (BLINK), FLINK                  ;1218
                63          50  7D 0016A          MOVQ    FLINK, (DISPTR)                 ;1223
            04  A0          53  D0 0016D          MOVL    DISPTR, 4(FLINK)                ;1225
                61          53  D0 00171          MOVL    DISPTR, (BLINK)                 ;1226
                02          56  D1 00174 21$:     CMPL    R6, #2                          ;1234
                            0A  13 00177          BEQL    22$                             ;
```

```
                     03        56  D1 00179          CMPL    R6, #3                                  ; 1235
                               05  13 0017C          BEQL    22$                                     ;
                     04        56  D1 0017E          CMPL    R6, #4                                  ; 1236
                               07  12 00181          BNEQ    23$                                     ;
                               53  DD 00183 22$:     PUSHL   DISPTR                                  ; 1238
                0000V  CF      01  FB 00185          CALLS   #1, DBG$SCR_GENERATE_SCREEN             ;
                     50        53  D0 0018A 23$:     MOVL    DISPTR, R0                              ; 1243
                               04  00018D           RET                                             ; 1245

; Routine Size:  398 bytes,    Routine Base:  DBG$CODE + 0CD5
```

```
;  1123          1246  1 ROUTINE DBG$SCR_EMPTY_DISPLAY(DISPTR): NOVALUE =
;  1124          1247  1 !
;  1125          1248  1 ! FUNCTION
;  1126          1249  1 !       This routine empties a screen display of its contents, meaning that
;  1127          1250  1 !       all text lines in the display are deleted.  When this routine returns,
;  1128          1251  1 !       the specified Screen Display Entry is still around with all its attri-
;  1129          1252  1 !       butes intact, but all text lines in the display (i.e., all Display Line
;  1130          1253  1 !       Entries) have been deleted and released back to the memory pool.
;  1131          1254  1 !
;  1132          1255  1 ! INPUTS
;  1133          1256  1 !       DISPTR  - A pointer to the Screen Display Entry for the display whose
;  1134          1257  1 !                 contents are to be emptied.
;  1135          1258  1 !
;  1136          1259  1 ! OUTPUTS
;  1137          1260  1 !       NONE
;  1138          1261  1 !
;  1139          1262  1
;  1140          1263  2     BEGIN
;  1141          1264  2
;  1142          1265  2     MAP
;  1143          1266  2         DISPTR: REF DBG$DISP_ENTRY;     ! Pointer to Screen Display Entry
;  1144          1267  2
;  1145          1268  2     LOCAL
;  1146          1269  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to current Display Line Entry
;  1147          1270  2         FLINK: REF DBG$DLINE_ENTRY;     ! Pointer to next Display Line Entry
;  1148          1271  2
;  1149          1272  2
;  1150          1273  2
;  1151          1274  2     ! Clear out all scroll settings, all Display Line Entry pointers, and the
;  1152          1275  2     ! line count in the Screen Display Entry.
;  1153          1276  2     !
;  1154          1277  2     DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  1155          1278  2     DISPTR[DBG$L_DISP_START_LINE_PTR] = DISPTR[DBG$L_DISP_START_LINE_PTR];
;  1156          1279  2     DISPTR[DBG$L_DISP_END_LINE_PTR] = DISPTR[DBG$L_DISP_START_LINE_PTR];
;  1157          1280  2     DISPTR[DBG$L_DISP_WINDOW_PTR] = 0;
;  1158          1281  2     DISPTR[DBG$W_DISP_LINECNT] = 0;
;  1159          1282  2     DISPTR[DBG$W_DISP_DROW] = 1;
;  1160          1283  2     DISPTR[DBG$W_DISP_SCROLL] = 0;
;  1161          1284  2     DISPTR[DBG$V_DISP_INVSCR] = TRUE;
;  1162          1285  2
;  1163          1286  2
;  1164          1287  2     ! Then delete all Screen Display Line Entries by releasing them back to
;  1165          1288  2     ! the memory pool.
;  1166          1289  2     !
;  1167          1290  2     WHILE .DLEPTR NEQ DISPTR[DBG$L_DISP_START_LINE_PTR] DO
;  1168          1291  3         BEGIN
;  1169          1292  3         FLINK = .DLEPTR[DBG$L_DLINE_FLINK];
;  1170          1293  3         DBG$REL_MEMORY(.DLEPTR);
;  1171          1294  3         DLEPTR = .FLINK;
;  1172          1295  2         END;
;  1173          1296  2
;  1174          1297  2
;  1175          1298  2     ! Also release all Error Line Entries attached to the display.
;  1176          1299  2     !
;  1177          1300  2     DLEPTR = .DISPTR[DBG$L_DISP_ERROR_PTR];
;  1178          1301  2     DISPTR[DBG$L_DISP_ERROR_PTR] = 0;
;  1179          1302  2     WHILE .DLEPTR NEQ 0 DO
```

```
;  1180          1303  3          BEGIN
;  1181          1304  3          FLINK = .DLEPTR[DBG$L_DLINE_FLINK];
;  1182          1305  3          DBG$REL_MEMORY(.DLEPTR);
;  1183          1306  3          DLEPTR = .FLINK;
;  1184          1307  2          END;
;  1185          1308  2
;  1186          1309  2
;  1187          1310  2      ! The display contents are now empty, so we return.
;  1188          1311  2      !
;  1189          1312  2      RETURN;
;  1190          1313  2
;  1191          1314  1      END;


                                007C 00000 DBG$SCR_EMPTY_DISPLAY:
                                           .WORD   Save R2,R3,R4,R5,R6                          ; 1246
                    56 00000000G   00  9E 00002    MOVAB   DBG$REL_MEMORY, R6                   ;
                    52         04  AC  D0 00009    MOVL    DISPTR, R2                           ; 1277
                    55         20  A2  9E 0000D    MOVAB   32(R2), R5                           ;
                    53             65  D0 00011    MOVL    (R5), DLEPTR                         ;
                    65             55  D0 00014    MOVL    R5, (R5)                             ; 1278
                 24 A2             55  D0 00017    MOVL    R5, 36(R2)                           ; 1279
                               28  A2  D4 0001B    CLRL    40(R2)                               ; 1280
                               1E  A2  B4 0001E    CLRW    30(R2)                               ; 1281
                 18 A2             01  B0 00021    MOVW    #1, 24(R2)                           ; 1282
                               0C  A2  B4 00025    CLRW    12(R2)                               ; 1283
                 0A A2             02  88 00028    BISB2   #2, 10(R2)                           ; 1284
                    55             53  D1 0002C 1$:    CMPL    DLEPTR, R5                       ; 1290
                                   0D  13 0002F    BEQL    2$                                   ;
                    54             63  D0 00031    MOVL    (DLEPTR), FLINK                      ; 1292
                                   53  DD 00034    PUSHL   DLEPTR                               ; 1293
                    66             01  FB 00036    CALLS   #1, DBG$REL_MEMORY                   ;
                    53             54  D0 00039    MOVL    FLINK, DLEPTR                        ; 1294
                                   EE  11 0003C    BRB     1$                                   ; 1290
                    53         2C  A2  D0 0003E 2$:    MOVL    44(R2), DLEPTR                   ; 1300
                               2C  A2  D4 00042    CLRL    44(R2)                               ; 1301
                                   53  D5 00045 3$:    TSTL    DLEPTR                           ; 1302
                                   0D  13 00047    BEQL    4$                                   ;
                    54             63  D0 00049    MOVL    (DLEPTR), FLINK                      ; 1304
                                   53  DD 0004C    PUSHL   DLEPTR                               ; 1305
                    66             01  FB 0004E    CALLS   #1, DBG$REL_MEMORY                   ;
                    53             54  D0 00051    MOVL    FLINK, DLEPTR                        ; 1306
                                   EF  11 00054    BRB     3$                                   ; 1302
                                       04 00056 4$:    RET                                      ; 1314

; Routine Size:  87 bytes,    Routine Base:  DBG$CODE + 0E63
```

```
;1193     1315  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_CANDISP_CMD(VERB_NODE): NOVALUE =
;1194     1316  1 !
;1195     1317  1 ! FUNCTION
;1196     1318  1 !       This routine executes the CANCEL DISPLAY command.  It accepts the
;1197     1319  1 !       address of the Verb Node for the command as input and then extracts
;1198     1320  1 !       the command parameters and executes the command.  The command param-
;1199     1321  1 !       eters are either a list of displays to be cancelled, each represented
;1200     1322  1 !       by a Noun Node, or the /ALL qualifier, represented by having no Noun
;1201     1323  1 !       Nodes.
;1202     1324  1 !
;1203     1325  1 ! INPUTS
;1204     1326  1 !       VERB_NODE - A pointer to the Verb Node for the CANCEL DISPLAY command
;1205     1327  1 !                   to be executed.  The Verb Node along with its attached Noun
;1206     1328  1 !                   Nodes contains all information picked up during the parsing
;1207     1329  1 !                   of the command.
;1208     1330  1 !
;1209     1331  1 ! OUTPUTS
;1210     1332  1 !       NONE
;1211     1333  1 !
;1212     1334  1
;1213     1335  2     BEGIN
;1214     1336  2
;1215     1337  2     MAP
;1216     1338  2         VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to input Verb Node
;1217     1339  2
;1218     1340  2     LOCAL
;1219     1341  2         DISPTR: REF DBG$DISP_ENTRY,      ! Pointer to current Display Entry
;1220     1342  2         NOUN_NODE: REF DBG$NOUN_NODE;    ! Pointer to current Noun Node
;1221     1343  2
;1222     1344  2
;1223     1345  2
;1224     1346  2     ! If there are no Noun Nodes, then the /ALL qualifier was specified and all
;1225     1347  2     ! displays are to be cancelled.  Hence we loop to delete all entries on the
;1226     1348  2     ! Screen Display List until no such entries are left.  We then return.
;1227     1349  2     !
;1228     1350  2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
;1229     1351  2     IF .NOUN_NODE EQL 0
;1230     1352  2     THEN
;1231     1353  3         BEGIN
;1232     1354  3         WHILE .DBG$SCR_DISPLAY_LIST[0] NEQ DBG$SCR_DISPLAY_LIST DO
;1233     1355  4             BEGIN
;1234     1356  4             DISPTR = .DBG$SCR_DISPLAY_LIST[0];
;1235     1357  4             DBG$SCR_DELETE_DISPLAY(DISPTR[DBG$A_DISP_NAME]);
;1236     1358  3             END;
;1237     1359  3
;1238     1360  3         RETURN;
;1239     1361  2         END;
;1240     1362  2
;1241     1363  2
;1242     1364  2     ! Noun Nodes are present, one for each specified display to cancel.  Loop
;1243     1365  2     ! through the Noun Nodes and delete each corresponding display.  Then
;1244     1366  2     ! return.
;1245     1367  2     !
;1246     1368  2     WHILE .NOUN_NODE NEQ 0 DO
;1247     1369  3         BEGIN
;1248     1370  3         DBG$SCR_DELETE_DISPLAY(.NOUN_NODE[DBG$L_NOUN_VALUE]);
;1249     1371  3         NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
```

```
;  1250          1372  2          END;
;  1251          1373  2
;  1252          1374  2      RETURN;
;  1253          1375  2
;  1254          1376  1      END;


                                                001C 00000          .ENTRY  DBG$SCR_EXECUTE_CANDISP_CMD, Save R2,R3,R4  ; 1315
                                    54 00000000'  EF   9E 00002          MOVAB   DBG$SCR_DISPLAY_LIST, R4                    ;
                                    50        04  AC   D0 00009          MOVL    VERB_NODE, R0                               ; 1350
                                    53        08  A0   D0 0000D          MOVL    8(R0), NOUN_NODE                            ;
                                                  17   12 00011          BNEQ    3$                                          ; 1351
                                    50            64   9E 00013 1$:      MOVAB   DBG$SCR_DISPLAY_LIST, R0                    ; 1354
                                    50            64   D1 00016          CMPL    DBG$SCR_DISPLAY_LIST, R0                    ;
                                                  1C   13 00019          BEQL    4$                                          ;
                                    52            64   D0 0001B          MOVL    DBG$SCR_DISPLAY_LIST, DISPTR                ; 1356
                                              4C  A2   9F 0001E          PUSHAB  76(DISPTR)                                  ; 1357
                                 FCF9         CF   01   FB 0002          CALLS   #1, DBG$SCR_DELETE_DISPLAY                  ;
                                                  EB   11 00026          BRB     1$                                          ; 1354
                                                  0D   13 00028 2$:      BEQL    4$                                          ; 1368
                                                  63   DD 0002A 3$:      PUSHL   (NOUN_NODE)                                 ; 1370
                                 FCEE         CF   01   FB 0002C          CALLS   #1, DBG$SCR_DELETE_DISPLAY                  ;
                                    53        08  A3   D0 00031          MOVL    8(NOUN_NODE), NOUN_NODE                     ; 1371
                                                  F1   11 00035          BRB     2$                                          ; 1368
                                                     04 00037 4$:        RET                                                 ; 1376

; Routine Size:  56 bytes,    Routine Base:  DBG$CODE + 0EBA
```

```
;  1256     1377  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_CANWIND_CMD(VERB_NODE): NOVALUE =
;  1257     1378  1 !
;  1258     1379  1 ! FUNCTION
;  1259     1380  1 !       This routine executes the CANCEL WINDOW command.  It accepts the
;  1260     1381  1 !       address of the Verb Node for the command as input and then extracts
;  1261     1382  1 !       the command parameters and executes the command.  The command param-
;  1262     1383  1 !       eters are either a list of windows to be cancelled, each represented
;  1263     1384  1 !       by a Noun Node, or the /ALL qualifier, represented by having no Noun
;  1264     1385  1 !       Nodes.
;  1265     1386  1 !
;  1266     1387  1 ! INPUTS
;  1267     1388  1 !       VERB_NODE - A pointer to the Verb Node for the CANCEL WINDOW command
;  1268     1389  1 !                   to be executed.  The Verb Node along with its attached Noun
;  1269     1390  1 !                   Nodes contains all information picked up during the parsing
;  1270     1391  1 !                   of the command.
;  1271     1392  1 !
;  1272     1393  1 ! OUTPUTS
;  1273     1394  1 !       NONE
;  1274     1395  1 !
;  1275     1396  1
;  1276     1397  2     BEGIN
;  1277     1398  2
;  1278     1399  2     MAP
;  1279     1400  2         VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to input Verb Node
;  1280     1401  2
;  1281     1402  2     LOCAL
;  1282     1403  2         NOUN_NODE: REF DBG$NOUN_NODE,    ! Pointer to current Noun Node
;  1283     1404  2         WPTR: REF DBG$WINDOW_ENTRY;      ! Pointer to current Window Entry
;  1284     1405  2
;  1285     1406  2
;  1286     1407  2
;  1287     1408  2     ! If there are no Noun Nodes, then the /ALL qualifier was specified and all
;  1288     1409  2     ! windows are to be cancelled.  Hence we loop to delete all entries on the
;  1289     1410  2     ! Screen Window List until no such entries are left.  We then return.
;  1290     1411  2     !
;  1291     1412  2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
;  1292     1413  2     IF .NOUN_NODE EQL 0
;  1293     1414  2     THEN
;  1294     1415  3         BEGIN
;  1295     1416  3         WHILE .DBG$SCR_WINDOW_LIST[0] NEQ DBG$SCR_WINDOW_LIST DO
;  1296     1417  4             BEGIN
;  1297     1418  4             WPTR = .DBG$SCR_WINDOW_LIST[0];
;  1298     1419  4             DBG$SCR_DELETE_WINDOW(WPTR[DBG$A_WINDOW_NAME]);
;  1299     1420  3             END;
;  1300     1421  3
;  1301     1422  3         RETURN;
;  1302     1423  2         END;
;  1303     1424  2
;  1304     1425  2
;  1305     1426  2     ! Noun Nodes are present, one for each specified window to cancel.  Loop
;  1306     1427  2     ! through the Noun Nodes and delete each corresponding window.  Then
;  1307     1428  2     ! return.
;  1308     1429  2     !
;  1309     1430  2     WHILE .NOUN_NODE NEQ 0 DO
;  1310     1431  3         BEGIN
;  1311     1432  3         DBG$SCR_DELETE_WINDOW(.NOUN_NODE[DBG$L_NOUN_VALUE]);
;  1312     1433  3         NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
```

```
;  1313          1434  2          END;
;  1314          1435  2
;  1315          1436  2      RETURN;
;  1316          1437  2
;  1317          1438  1      END;


                                  001C 00000          .ENTRY  DBG$SCR_EXECUTE_CANWIND_CMD, Save R2,R3,R4 ; 1377
                        54 00000000'  EF   9E 00002          MOVAB   DBG$SCR_WINDOW_LIST, R4                    ;
                        50        04  AC   D0 00009          MOVL    VERB_NODE, R0                              ; 1412
                        53        08  A0   D0 0000D          MOVL    8(R0), NOUN_NODE                           ;
                                      17   12 00011          BNEQ    3$                                         ; 1413
                        50            64   9E 00013 1$:      MOVAB   DBG$SCR_WINDOW_LIST, R0                    ; 1416
                        50            64   D1 00016          CMPL    DBG$SCR_WINDOW_LIST, R0                    ;
                                      1C   13 00019          BEQL    4$                                         ;
                        52            64   D0 0001B          MOVL    DBG$SCR_WINDOW_LIST, WPTR                  ; 1418
                                10    A2   9F 0001E          PUSHAB  16(WPTR)                                   ; 1419
                    FD94  CF          01   FB 00021          CALLS   #1, DBG$SCR_DELETE_WINDOW                  ;
                                      EB   11 00026          BRB     1$                                         ; 1416
                                      0D   13 00028 2$:      BEQL    4$                                         ; 1430
                                      63   DD 0002A 3$:      PUSHL   (NOUN_NODE)                                ; 1432
                    FD89  CF          01   FB 0002C          CALLS   #1, DBG$SCR_DELETE_WINDOW                  ;
                        53        08  A3   D0 00031          MOVL    8(NOUN_NODE), NOUN_NODE                    ; 1433
                                      F1   11 00035          BRB     2$                                         ; 1430
                                      04   00037 4$:         RET                                                ; 1438

; Routine Size:  56 bytes,    Routine Base:  DBG$CODE + 0EF2
```

```
;  1319          1439  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_DISPLAY_CMD(VERB_NODE, SET_FLAG): NOVALUE =
;  1320          1440  1 !
;  1321          1441  1 ! FUNCTION
;  1322          1442  1 !       This routine executes the DISPLAY and SET DISPLAY commands.  It
;  1323          1443  1 !       accepts the address of a Verb Node for the command as input and
;  1324          1444  1 !       it extracts the command parameters so that the command can be
;  1325          1445  1 !       executed.  Since DISPLAY and SET DISPLAY are very similar commands
;  1326          1446  1 !       in their semantic effects, most processing for these two commands
;  1327          1447  1 !       is identical, with the differences special-cased as necessary.
;  1328          1448  1 !
;  1329          1449  1 ! INPUTS
;  1330          1450  1 !       VERB_NODE - A pointer to the Verb Node for the command to be executed.
;  1331          1451  1 !                   The Verb Node, along with its attached Adverb and Noun
;  1332          1452  1 !                   Nodes, contains all information picked up during the
;  1333          1453  1 !                   parsing of the command.
;  1334          1454  1 !
;  1335          1455  1 !       SET_FLAG - TRUE if the command to be executed is the SET DISPLAY
;  1336          1456  1 !                  command and FALSE if the command is the DISPLAY command.
;  1337          1457  1 !
;  1338          1458  1 ! OUTPUTS
;  1339          1459  1 !       NONE
;  1340          1460  1 !
;  1341          1461  1
;  1342          1462  2     BEGIN
;  1343          1463  2
;  1344          1464  2     MAP
;  1345          1465  2         VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to input Verb Node
;  1346          1466  2
;  1347          1467  2     LOCAL
;  1348          1468  2         ADVERB_NODE:                     ! Pointer to the Adverb Node with the
;  1349          1469  2                 REF DBG$ADVERB_NODE,     !      command qualifier information
;  1350          1470  2         CLEAR_FLAG,                      ! Flag set for /CLEAR qualifier
;  1351          1471  2         CONTENT_PTR,                     ! Pointer to DEBUG command list which
;  1352          1472  2                                          !      defines display contents
;  1353          1473  2         DISPTR: REF DBG$DISP_ENTRY,      ! Pointer to current Display Entry
;  1354          1474  2         FLAGWORD,                        ! Flag & size longword from Adverb Node
;  1355          1475  2         GENERATE_ALL_FLAG,               ! Flag set for /GENERATE all displays
;  1356          1476  2         GENERATE_FLAG,                   ! Flag set for /GENERATE qualifier
;  1357          1477  2         HIDE_FLAG,                       ! Flag set for /HIDE qualifier
;  1358          1478  2         KIND,                            ! The display kind (content spec kind)
;  1359          1479  2         LIST_LOCATION,                   ! Display list location (for /HIDE)
;  1360          1480  2         MARK_FLAG,                       ! Flag set for /MARK_CHANGE qualifier
;  1361          1481  2         NOMARK_FLAG,                     ! Flag set for /NOMARK_CHANGE qualifier
;  1362          1482  2         NAMEPTR: REF VECTOR[,BYTE],      ! Pointer to ASCIC display name
;  1363          1483  2         NOUN_NODE: REF DBG$NOUN_NODE,    ! Pointer to current Noun Node with
;  1364          1484  2                                          !      display and attribute info
;  1365          1485  2         REFRESH_FLAG,                    ! Flag set for DISPLAY/REFRESH command
;  1366          1486  2         REMOVED_FLAG,                    ! Flag set for /REMOVED qualifier
;  1367          1487  2         SIZE,                            ! Size of display (maximum lines)
;  1368          1488  2         TEMP,                            ! Parameter for SMG$SET_KEYPAD_MODE
;  1369          1489  2         WPTR;                            ! Pointer to current Screen Window Entry
;  1370          1490  2
;  1371          1491  2
;  1372          1492  2
;  1373          1493  2     ! Extract all the qualifier information from the command Adverb Node.
;  1374          1494  2     !
;  1375          1495  2     ADVERB_NODE = .VERB_NODE[DBG$L_VERB_ADVERB_PTR];
```

```
;  1376          1496  2      FLAGWORD = .ADVERB_NODE[DBG$L_ADVERB_VALUE];
;  1377          1497  2      REFRESH_FLAG = .FLAGWORD<V_(0)>;
;  1378          1498  2      CLEAR_FLAG = .FLAGWORD<V_(1)>;
;  1379          1499  2      GENERATE_ALL_FLAG = .FLAGWORD<V_(2)>;
;  1380          1500  2      GENERATE_FLAG = .FLAGWORD<V_(3)>;
;  1381          1501  2      HIDE_FLAG = .FLAGWORD<V_(4)>;
;  1382          1502  2      MARK_FLAG = .FLAGWORD<V_(5)>;
;  1383          1503  2      NOMARK_FLAG = .FLAGWORD<V_(6)>;
;  1384          1504  2      REMOVED_FLAG = .FLAGWORD<V_(7)>;
;  1385          1505  2      SIZE = .FLAGWORD<V_(16,16)>;
;  1386          1506  2
;  1387          1507  2
;  1388          1508  2      ! If this is the DISPLAY/REFRESH command, process that command right away.
;  1389          1509  2      ! There are no Noun Nodes for this command.  Note that we reset "applica-
;  1390          1510  2      ! tion mode" on the terminal (activate the keypad keys) if keypad mode is
;  1391          1511  2      ! set.
;  1392          1512  2      !
;  1393          1513  2      IF .REFRESH_FLAG
;  1394          1514  2      THEN
;  1395          1515  3          BEGIN
;  1396          1516  3          INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  1397          1517  3              OLD_VALID[.I] = FALSE;
;  1398          1518  3
;  1399          1519  3          IF .DBG$GB_KEYPAD_INPUT
;  1400          1520  3          THEN
;  1401          1521  4              BEGIN
;  1402          1522  4              TEMP = 1;
;  1403          1523  4              SMG$SET_KEYPAD_MODE(DBG$GL_KEYBOARD_ID, TEMP);
;  1404          1524  3              END;
;  1405          1525  3
;  1406          1526  3          RETURN;
;  1407          1527  2          END;
;  1408          1528  2
;  1409          1529  2
;  1410          1530  2      ! If this is the DISPLAY/GENERATE command without a display list, we also
;  1411          1531  2      ! process the command right away as there are no Noun Nodes.  Here we
;  1412          1532  2      ! regenerate the contents of all non-removed, automatically updated screen
;  1413          1533  2      ! displays.
;  1414          1534  2      !
;  1415          1535  2      IF .GENERATE_ALL_FLAG
;  1416          1536  2      THEN
;  1417          1537  3          BEGIN
;  1418          1538  3          DBG$SCR_GENERATE_SCREEN(0);
;  1419          1539  3          RETURN;
;  1420          1540  2          END;
;  1421          1541  2
;  1422          1542  2
;  1423          1543  2      ! If this is the SET DISPLAY command and no display size was specified
;  1424          1544  2      ! use the default display size.
;  1425          1545  2      !
;  1426          1546  2      IF .SET_FLAG AND (.SIZE EQL 0) THEN SIZE = 50;
;  1427          1547  2
;  1428          1548  2
;  1429          1549  2      ! Loop through all displays specified on the command and take the appro-
;  1430          1550  2      ! priate semantic action for each such display.
;  1431          1551  2      !
;  1432          1552  2      NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
```

```
;  1433          1553  2     WHILE .NOUN_NODE NEQ 0 DO
;  1434          1554  3         BEGIN
;  1435          1555  3
;  1436          1556  3
;  1437          1557  3         ! Extract all the parameter values from the Noun Node.
;  1438          1558  3         !
;  1439          1559  3         NAMEPTR = .NOUN_NODE[DBG$L_NOUN_VALUE];
;  1440          1560  3         WPTR = .NOUN_NODE[DBG$L_NOUN_VALUE2];
;  1441          1561  3         KIND = .NOUN_NODE[DBG$L_NOUN_VALUE3];
;  1442          1562  3         CONTENT_PTR = .NOUN_NODE[DBG$L_NOUN_VALUE4];
;  1443          1563  3
;  1444          1564  3
;  1445          1565  3         ! Refetch the mark-changes and generate flags--these flags are modified
;  1446          1566  3         ! in the loop and must therefore be reset here.
;  1447          1567  3         !
;  1448          1568  3         GENERATE_FLAG = .FLAGWORD<V_(3)>;
;  1449          1569  3         MARK_FLAG = .FLAGWORD<V_(5)>;
;  1450          1570  3
;  1451          1571  3
;  1452          1572  3         ! Set the list location in the Screen Display List where the display
;  1453          1573  3         ! should be inserted.  This is determined by the /HIDE qualifier.
;  1454          1574  3         !
;  1455          1575  3         LIST_LOCATION = +1;
;  1456          1576  3         IF .HIDE_FLAG THEN LIST_LOCATION = -1;
;  1457          1577  3
;  1458          1578  3
;  1459          1579  3         ! If this is a SET DISPLAY command, call CREATE_DISPLAY to create the
;  1460          1580  3         ! desired new display.
;  1461          1581  3         !
;  1462          1582  3         IF .SET_FLAG
;  1463          1583  3         THEN
;  1464          1584  3             DISPTR = DBG$SCR_CREATE_DISPLAY  NAMEPTR, .KIND,
;  1465          1585  3                                              .WPTR, .SIZE, .CONTENT_PTR)
;  1466          1586  3
;  1467          1587  3
;  1468          1588  3         ! Otherwise, this is a DISPLAY command, so we call DISPLAY_COMMAND to
;  1469          1589  3         ! execute the desired semantics on the specified display.
;  1470          1590  3         !
;  1471          1591  3         ELSE
;  1472          1592  3             DISPTR = DBG$SCR_DISPLAY_COMMAND(.NAMEPTR, .KIND, .WPTR,
;  1473          1593  3                                              .SIZE, .CONTENT_PTR,   IST_LOCATION);
;  1474          1594  3
;  1475          1595  3
;  1476          1596  3         ! If this display was marked as removed but is being "unremoved" with
;  1477          1597  3         ! this DISPLAY command, set the generate flag.  This ensures that an
;  1478          1598  3         ! automatically updated display has its contents regenerated whenever
;  1479          1599  3         ! it is unremoved.
;  1480          1600  3         !
;  1481          1601  3         IF .DISPTR[DBG$V_DISP_REMOVE] AND (NOT .REMOVED_FLAG)
;  1482          1602  3         THEN
;  1483          1603  3             GENERATE_FLAG = TRUE;
;  1484          1604  3
;  1485          1605  3
;  1486          1606  3         ! If this display kind is not (or is no longer) DBG$K_DISP_DO, clear
;  1487          1607  3         ! the mark-change flag in the display entry.  If the /MARK_CHANGE or
;  1488          1608  3         ! /NOMARK_CHANGE qualifier was also specified on this command, signal
;  1489          1609  3         ! an informational message.
```

```
;  1490     1610  3          !
;  1491     1611  3          IF .DISPTR[DBG$B_DISP_KIND] NEQ DBG$K_DISP_DO
;  1492     1612  3          THEN
;  1493     1613  4              BEGIN
;  1494     1614  4              DISPTR[DBG$V_DISP_MARKFLG] = FALSE;
;  1495     1615  4              IF .MARK_FLAG OR .NOMARK_FLAG
;  1496     1616  4              THEN
;  1497     1617  4                  SIGNAL(DBG$_NOMARKCHNG, 1, DISPTR[DBG$A_DISP_NAME]);
;  1498     1618  4
;  1499     1619  4              MARK_FLAG = FALSE;
;  1500     1620  3              END;
;  1501     1621  3
;  1502     1622  3
;  1503     1623  3          ! Set the remove and mark-changes flags as appropriate.
;  1504     1624  3          !
;  1505     1625  3          DISPTR[DBG$V_DISP_REMOVE] = .REMOVED_FLAG;
;  1506     1626  3          IF .MARK_FLAG THEN DISPTR[DBG$V_DISP_MARKFLG] = TRUE;
;  1507     1627  3          IF .NOMARK_FLAG THEN DISPTR[DBG$V_DISP_MARKFLG] = FALSE;
;  1508     1628  3
;  1509     1629  3
;  1510     1630  3          ! If the display is being removed, remove any permanent pointers to
;  1511     1631  3          ! the display's Screen Display Entry.  This means that any SELECT
;  1512     1632  3          ! setting to this display is cancelled.
;  1513     1633  3          !
;  1514     1634  3          IF .REMOVED_FLAG
;  1515     1635  3          THEN
;  1516     1636  4              BEGIN
;  1517     1637  4              IF .DISPTR EQL .DBG$SCR_CURDISP_INPUT  THEN DBG$SCR_CURDISP_INPUT  = 0;
;  1518     1638  4              IF .DISPTR EQL .DBG$SCR_CURDISP_OUTPUT THEN DBG$SCR_CURDISP_OUTPUT = 0;
;  1519     1639  4              IF .DISPTR EQL .DBG$SCR_CURDISP_SCROLL THEN DBG$SCR_CURDISP_SCROLL = 0;
;  1520     1640  4              IF .DISPTR EQL .DBG$SCR_CURDISP_SOURCE THEN DBG$SCR_CURDISP_SOURCE = 0;
;  1521     1641  4              IF .DISPTR EQL .DBG$GL_SCREEN_ERROR   THEN DBG$GL_SCREEN_ERROR   = 0;
;  1522     1642  4              IF .DISPTR EQL .DBG$GL_SCREEN_HISTORY THEN DBG$GL_SCREEN_HISTORY = 0;
;  1523     1643  4              IF .DISPTR EQL .DBG$GL_SCREEN_INPUT   THEN DBG$GL_SCREEN_INPUT   = 0;
;  1524     1644  4              IF .DISPTR EQL .DBG$GL_SCREEN_OUTPUT  THEN DBG$GL_SCREEN_OUTPUT  = 0;
;  1525     1645  4              IF .DISPTR EQL .DBG$GL_SCREEN_SOURCE  THEN DBG$GL_SCREEN_SOURCE  = 0;
;  1526     1646  4              DISPTR[DBG$V_DISP_INVSCR] = TRUE;
;  1527     1647  3              END;
;  1528     1648  3
;  1529     1649  3
;  1530     1650  3          ! If the /CLEAR qualifier was specified, clear the display's entire
;  1531     1651  3          ! contents so there are no lines of text in the display.  Also clear
;  1532     1652  3          ! out all SOURCE display module context information.
;  1533     1653  3          !
;  1534     1654  3          IF .CLEAR_FLAG
;  1535     1655  3          THEN
;  1536     1656  4              BEGIN
;  1537     1657  4              DBG$SCR_EMPTY_DISPLAY(.DISPTR);
;  1538     1658  4              DISPTR[DBG$L_DISP_MODPTR] = 0;
;  1539     1659  4              DISPTR[DBG$L_DISP_CENTER] = 0;
;  1540     1660  4              DISPTR[DBG$L_DISP_MARKLINE] = 0;
;  1541     1661  4              DISPTR[DBG$L_DISP_MINLINE] = 0;
;  1542     1662  4              DISPTR[DBG$L_DISP_MAXLINE] = 0;
;  1543     1663  3              END;
;  1544     1664  3
;  1545     1665  3
;  1546     1666  3          ! If this is the SET DISPLAY command or if the /GENERATE qualifier was
```

```
;  1547          1667  3          ! specified on the DISPLAY command, generate the display contents for
;  1548          1668  3          ! automatically generated displays.
;  1549          1669  3          !
;  1550          1670  3          IF .SET_FLAG OR .GENERATE_FLAG THEN DBG$SCR_GENERATE_SCREEN(.DISPTR);
;  1551          1671  3
;  1552          1672  3
;  1553          1673  3          ! Link to the next Noun Node and loop.
;  1554          1674  3          !
;  1555          1675  3          NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
;  1556          1676  3
;  1557          1677  2          END;                                ! End of loop over noun nodes (displays)
;  1558          1678  2
;  1559          1679  2
;  1560          1680  2      ! The command processing is completed.  Now return.
;  1561          1681  2      !
;  1562          1682  2      RETURN;
;  1563          1683  2
;  1564          1684  1      END;


                                        0FFC 00000          .ENTRY  DBG$SCR_EXECUTE_DISPLAY_CMD, Save R2,R3,R4,-  ; 1439
                                                                    R5,R6,R7,R8,R9,R10,R11                            :
                                    5E          10  C2 00002        SUBL2   #16, SP                                   :
                                    52    04    AC  D0 00005        MOVL    VERB_NODE, R2                             ; 1495
                                    50    04    A2  D0 00009        MOVL    4(R2), ADVERB_NODE                        :
                                    55    04    A0  D0 0000D        MOVL    4(ADVERB_NODE), FLAGWORD                  ; 1496
                50                  55          01  00 EF 00011    EXTZV   #0, #1, FLAGWORD, REFRESH_FLAG            ; 1497
                7E                  55          01  01 EF 00016    EXTZV   #1, #1, FLAGWORD, CLEAR_FLAG              ; 1498
                53                  55          01  02 EF 0001B    EXTZV   #2, #1, FLAGWORD, GENERATE_ALL_FLAG       ; 1499
                58                  55          01  03 EF 00020    EXTZV   #3, #1, FLAGWORD, GENERATE_FLAG           ; 1500
                7E                  55          01  04 EF 00025    EXTZV   #4, #1, FLAGWORD, HIDE_FLAG               ; 1501
                57                  55          01  05 EF 0002A    EXTZV   #5, #1, FLAGWORD, MARK_FLAG               ; 1502
                7E                  55          01  06 EF 0002F    EXTZV   #6, #1, FLAGWORD, NOMARK_FLAG             ; 1503
                5A                  55          01  07 EF 00034    EXTZV   #7, #1, FLAGWORD, REMOVED_FLAG            ; 1504
                56                  55          10  10 EF 00039    EXTZV   #16, #16, FLAGWORD, SIZE                  ; 1505
                                            2B          50 E9 0003E    BLBC    REFRESH_FLAG, 4$                      ; 1513
                                                        50 D4 00041    CLRL    I                                     ; 1516
                                    00 00000000'    EF      50 E5 00043 1$:    BBCC    I, OLD_VALID, 2$              ; 1517
                                    F4              50      14 F3 00048 2$:    AOBLEQ  #20, I, 1$                    :
                                                01  00000000G 00 E8 0004F    BLBS    DBG$GB_KEYPAD_INPUT, 3$           ; 1519
                                                            04 00056        RET                                       :
                                            18      AE          01  D0 00057 3$:    MOVL    #1, TEMP                  ; 1522
                                                        18      AE  9F 0005B    PUSHAB  TEMP                          ; 1523
                                                00000000G   00      9F 0005E    PUSHAB  DBG$GL_KEYBOARD_ID            :
                                        00000000G   00              02 FB 00064    CALLS   #2, SMG$SET_KEYPAD_MODE   :
                                                                    04 0006B    RET                                   ; 1515
                                                08              53 E9 0006C 4$:    BLBC    GENERATE_ALL_FLAG, 5$     ; 1535
                                                                7E D4 0006F    CLRL    -(SP)                         ; 1538
                                            0000V   CF          01 FB 00071    CALLS   #1, DBG$SCR_GENERATE_SCREEN   :
                                                                    04 00076    RET                                   ; 1537
                                                05      08      AC E9 00077 5$:    BLBC    SET_FLAG, 6$              ; 1546
                                                                03 12 0007B    BNEQ    6$                            :
                                                56              32 D0 0007D    MOVL    #50, SIZE                     :
                                                53      0E      A2 D0 00080 6$:    MOVL    8(R2), NOUN_NODE          ; 1552
```

```
                                                01  12 00084 7$:     BNEQ    8$                                              ; 1553
                                                    04 00086          RET
                                             5B     63  D0 00087 8$:     MOVL    (NOUN_NODE), NAMEPTR                            ; 1559
                                             59  0C A3  D0 0008A          MOVL    12(NOUN_NODE), WPTR                             ; 1560
                                          10 AE  10 A3  7D 0008E          MOVQ    16(NOUN_NODE), KIND                             ; 1561
                                    58    55       01  03  EF 00093          EXTZV   #3, #1, FLAGWORD, GENERATE_FLAG         ; 1568
                                    57    55       01  05  EF 00098          EXTZV   #5, #1, FLAGWORD, MARK_FLAG             ; 1569
                                          0C AE    01  D0 0009D          MOVL    #1, LIST_LOCATION                               ; 1575
                                             04  04 AE  E9 000A1          BLBC    HIDE_FLAG, 9$                                   ; 1576
                                          0C AE    01  CE 000A5          MNEGL   #1, LIST_LOCATION
                                             13  08 AC  E9 000A9 9$:     BLBC    SET_FLAG, 10$                                   ; 1584
                                                 14 AE  DD 000AD          PUSHL   CONTENT_PTR                                     ; 1585
                                                    56  DD 000B0          PUSHL   SIZE
                                                    59  DD 000B2          PUSHL   WPTR
                                                 1C AE  DD 000B4          PUSHL   KIND                                            ; 1584
                                                    5B  DD 000B7          PUSHL   NAMEPTR
                                       FA8E     CF      05  FB 000B9          CALLS   #5, DBG$SCR_CREATE_DISPLAY
                                                    14  11 000BE          BRB     11$
                                                 0C AE  DD 000C0 10$:     PUSHL   LIST_LOCATION                                   ; 1593
                                                 18 AE  DD 000C3          PUSHL   CONTENT_PTR
                                                    56  DD 000C6          PUSHL   SIZE
                                                    59  DD 000C8          PUSHL   WPTR                                            ; 1592
                                                 20 AE  DD 000CA          PUSHL   KIND
                                                    5B  DD 000CD          PUSHL   NAMEPTR
                                       FCD7     CF      06  FB 000CF          CALLS   #6, DBG$SCR_DISPLAY_COMMAND
                                             52      50  D0 000D4 11$:     MOVL    R0, DISPTR
                                             54  08 A2  9E 000D7          MOVAB   8(DISPTR), R4                                   ; 1601
                                             06  02 A4  E9 000DB          BLBC    2(R4), 12$
                                             03      5A  E8 000DF          BLBS    REMOVED_FLAG, 12$
                                             58      01  D0 000E2          MOVL    #1, GENERATE_FLAG                               ; 1603
                                             02      64  91 000E5 12$:     CMPB    (R4), #2                                        ; 1611
                                                     1E  13 000E8          BEQL    15$
                                          02 A4          10  8A 000EA          BICB2   #16, 2(R4)                                      ; 1614
                                             03          57  E8 000EE          BLBS    MARK_FLAG, 13$                                  ; 1615
                                             12          6E  E9 000F1          BLBC    NOMARK_FLAG, 14$
                                                    4C   A2  9F 000F4 13$:     PUSHAB  76(DISPTR)                                      ; 1617
                                                         01  DD 000F7          PUSHL   #1
                                                00028FD3 8F  DD 000F9          PUSHL   #167891
                                    00000000G    00      03  FB 000FF          CALLS   #3, LIB$SIGNAL
                                                         57  D4 00106 14$:     CLRL    MARK_FLAG                                       ; 1619
                                    02 A4     01         00  F0 00108 15$:     INSV    REMOVED_FLAG, #0, #1, 2(R4)                     ; 1625
                                                         04  E9 0010E          BLBC    MARK_FLAG, 16$                                  ; 1626
                                          02 A4          10  88 00111          BISB2   #16, 2(R4)
                                             04          6E  E9 00115 16$:     BLBC    NOMARK_FLAG, 17$                                ; 1627
                                          02 A4          10  8A 00118          BICB2   #16, 2(R4)
                                             03          5A  E8 0011C 17$:     BLBS    REMOVED_FLAG, 18$                               ; 1634
                                                     008B 31  0011F          BRW     28$
                                    00000000'    EF      52  D1 00122 18$:     CMPL    DISPTR, DBG$SCR_CURDISP_INPUT                   ; 1637
                                                         06  12 00129          BNEQ    19$
                                                00000000' EF  D4 0012B          CLRL    DBG$SCR_CURDISP_INPUT
                                    00000000'    EF      52  D1 00131 19$:     CMPL    DISPTR, DBG$SCR_CURDISP_OUTPUT                  ; 1638
                                                         06  12 00138          BNEQ    20$
                                                00000000' EF  D4 0013A          CLRL    DBG$SCR_CURDISP_OUTPUT
                                    00000000'    EF      52  D1 00140 20$:     CMPL    DISPTR, DBG$SCR_CURDISP_SCROLL                  ; 1639
                                                         06  12 00147          BNEQ    21$
                                                00000000' EF  D4 00149          CLRL    DBG$SCR_CURDISP_SCROLL
                                    00000000'    EF      52  D1 0014F 21$:     CMPL    DISPTR, DBG$SCR_CURDISP_SOURCE                  ; 1640
```

```
                                    06   12 00156          BNEQ    22$                                                    ;
                          00000000' EF   D4 00158          CLRL    DBG$SCR_CURDISP_SOURCE                                 ;
00000000'  EF                       52   D1 0015E 22$:     CMPL    DISPTR, DBG$GL_SCREEN_ERROR                            ; 1641
                                    06   12 00165          BNEQ    23$                                                    ;
                          00000000' EF   D4 00167          CLRL    DBG$GL_SCREEN_ERROR                                    ;
00000000'  EF                       52   D1 0016D 23$:     CMPL    DISPTR, DBG$GL_SCREEN_HISTORY                          ; 1642
                                    06   12 00174          BNEQ    24$                                                    ;
                          00000000' EF   D4 00176          CLRL    DBG$GL_SCREEN_HISTORY                                  ;
00000000'  EF                       52   D1 0017C 24$:     CMPL    DISPTR, DBG$GL_SCREEN_INPUT                            ; 1643
                                    06   12 00183          BNEQ    25$                                                    ;
                          00000000' EF   D4 00185          CLRL    DBG$GL_SCREEN_INPUT                                    ;
00000000'  EF                       52   D1 0018B 25$:     CMPL    DISPTR, DBG$GL_SCREEN_OUTPUT                           ; 1644
                                    06   12 00192          BNEQ    26$                                                    ;
                          00000000' EF   D4 00194          CLRL    DBG$GL_SCREEN_OUTPUT                                   ;
00000000'  EF                       52   D1 0019A 26$:     CMPL    DISPTR, DBG$GL_SCREEN_SOURCE                           ; 1645
                                    06   12 001A1          BNEQ    27$                                                    ;
                          00000000' EF   D4 001A3          CLRL    DBG$GL_SCREEN_SOURCE                                   ;
          02   A4                   02   88 001A9 27$:     BISB2   #2, 2(R4)                                              ; 1646
          10        08   AE         E9   AD 001AD 28$:     BLBC    CLEAR_FLAG, 29$                                        ; 1654
                                    52   DD 001B1          PUSHL   DISPTR                                                 ; 1657
        FD81   CF                   01   FB 001B3          CALLS   #1, DBG$SCR_EMPTY_DISPLAY                              ;
                    34   A2         7C   00 001B8          CLRQ    52(DISPTR)                                             ; 1658
                    3C   A2         7C   00 001BB          CLRQ    60(DISPTR)                                             ; 1660
                    44   A2         D4   00 001BE          CLRL    68(DISPTR)                                             ; 1662
          03        08   AC         E8   AD 001C1 29$:     BLBS    SET_FLAG, 30$                                          ; 1670
          07             58         E9   AD 001C5          BLBC    GENERATE_FLAG, 31$                                     ;
                                    52   DD 001C8 30$:     PUSHL   DISPTR                                                 ;
        0000V  CF                   01   FB 001CA          CALLS   #1, DBG$SCR_GENERATE_SCREEN                            ;
          53        08   A3         D0   00 001CF 31$:     MOVL    8(NOUN_NODE), NOUN_NODE                                ; 1675
                         FEAE       31   00 001D3          BRW     7$                                                     ; 1553
                                    04   00 001D6          RET                                                            ; 1684

; Routine Size:  471 bytes,    Routine Base:  DBG$CODE + 0F2A
```

```
;1566     1685  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_SAVE_CMD(VERB_NODE): NOVALUE =
;1567     1686  1 !
;1568     1687  1 ! FUNCTION
;1569     1688  1 !       This routine executes the SAVE command which saves the contents of an
;1570     1689  1 !       existing display in a new display with a specified name.  It accepts
;1571     1690  1 !       the address of a Verb Node for a SAVE command as input and it extracts
;1572     1691  1 !       the command parameters so that the command can be executed.  To execute
;1573     1692  1 !       the SAVE command, a new display of kind NORMAL or SOURCE is created,
;1574     1693  1 !       after which the entire contents of the original display is copied to
;1575     1694  1 !       the new display.  The new display also inherits the window attributes
;1576     1695  1 !       and scrolling position of the original display.
;1577     1696  1 !
;1578     1697  1 ! INPUTS
;1579     1698  1 !       VERB_NODE - A pointer to the Verb Node for the command to be executed.
;1580     1699  1 !                   The Verb Node, along with its attached Noun Nodes, contains
;1581     1700  1 !                   all information picked up during the parsing of the command.
;1582     1701  1 !
;1583     1702  1 ! OUTPUTS
;1584     1703  1 !       NONE
;1585     1704  1 !
;1586     1705  1
;1587     1706  2     BEGIN
;1588     1707  2
;1589     1708  2     MAP
;1590     1709  2         VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to the command Verb Node
;1591     1710  2
;1592     1711  2     LOCAL
;1593     1712  2         DISPTR: REF DBG$DISP_ENTRY,      ! Pointer to existing Display Entry
;1594     1713  2         DIS_BLINK: REF DBG$DISP_ENTRY,   ! Pointer to previous Display Entry
;1595     1714  2         DIS_FLINK: REF DBG$DISP_ENTRY,   ! Pointer to next Display Entry
;1596     1715  2         DLEPTR: REF DBG$DLINE_ENTRY,     ! Pointer to existing Display Line Entry
;1597     1716  2         DL_BLINK: REF DBG$DLINE_ENTRY,   ! Pointer to previous Display Line Entry
;1598     1717  2         DL_FLINK: REF DBG$DLINE_ENTRY,   ! Pointer to next Display Line Entry
;1599     1718  2         HANDLER_DISPTR: VOLATILE,        ! New Display Entry pointer for handler
;1600     1719  2         NAMEPTR: REF VECTOR[,BYTE],      ! Pointer to new display name
;1601     1720  2         NEWDISP: REF DBG$DISP_ENTRY,     ! Pointer to new Screen Display Entry
;1602     1721  2         NEWDLE: REF DBG$DLINE_ENTRY,     ! Pointer to new Display Line Entry
;1603     1722  2         NOUN_NODE: REF DBG$NOUN_NODE,    ! Pointer to the current Noun Node
;1604     1723  2         TEXTCNT,                         ! Number of text characters to copy
;1605     1724  2         TEXTPTR: REF VECTOR[,BYTE];      ! Pointer to display line's text
;1606     1725  2
;1607     1726  2     ENABLE
;1608     1727  2         HANDLER_EXECUTE_SAVE(            ! Set up a handler for NOFREE condition
;1609     1728  2                              HANDLER_DISPTR);!     and pass in new Display pointer
;1610     1729  2
;1611     1730  2
;1612     1731  2
;1613     1732  2     ! Loop through all the Noun Nodes attached to the Verb Node.  Each Node
;1614     1733  2     ! Node corresponds to a SAVE specification to be executed.
;1615     1734  2     !
;1616     1735  2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
;1617     1736  2     WHILE .NOUN_NODE NEQ 0 DO
;1618     1737  3         BEGIN
;1619     1738  3
;1620     1739  3
;1621     1740  3         ! See if a display already exists that has the same name as the speci-
;1622     1741  3         ! fied new display.  If it does, we signal an error and do not allow
```

```
;  1623     1742  3          ! the new display to be created.
;  1624     1743  3          !
;  1625     1744  3          DISPTR = DBG$SCR_LOOKUP_DISPLAY(.NOUN_NODE[DBG$L_NOUN_VALUE2]);
;  1626     1745  3          IF .DISPTR NEQ 0
;  1627     1746  3          THEN
;  1628     1747  3              SIGNAL(DBG$_DISPEXISTS, 1, .NOUN_NODE[DBG$L_NOUN_VALUE2]);
;  1629     1748  3
;  1630     1749  3
;  1631     1750  3          ! Pick up the parameters for the current SAVE specification and allo-
;  1632     1751  3          ! cate a Screen Display Entry for the new display to be created.
;  1633     1752  3          !
;  1634     1753  3          DISPTR = .NOUN_NODE[DBG$L_NOUN_VALUE];
;  1635     1754  3          NAMEPTR = .NOUN_NODE[DBG$L_NOUN_VALUE2];
;  1636     1755  3          HANDLER_DISPTR = 0;
;  1637     1756  3          NEWDISP = DBG$GET_MEMORY(DBG$K_DISP_ENTSIZE + .NAMEPTR[0]/%UPVAL + 1);
;  1638     1757  3          HANDLER_DISPTR = .NEWDISP;
;  1639     1758  3
;  1640     1759  3
;  1641     1760  3          ! Fill the display name and the display kind into the new Screen
;  1642     1761  3          ! Display Entry.  Note that the kind becomes NORMAL unless this is
;  1643     1762  3          ! a source display and that the DEBUG Command List pointer is
;  1644     1763  3          ! always left zeroed out.
;  1645     1764  3          !
;  1646     1765  3          CH$MOVE(.NAMEPTR[0] + 1, NAMEPTR[0], NEWDISP[DBG$A_DISP_NAME]);
;  1647     1766  3          IF .DISPTR[DBG$B_DISP_KIND] EQL DBG$K_DISP_SOURCE
;  1648     1767  3          THEN
;  1649     1768  3              NEWDISP[DBG$B_DISP_KIND] = DBG$K_DISP_SOURCE
;  1650     1769  3
;  1651     1770  3          ELSE
;  1652     1771  3              NEWDISP[DBG$B_DISP_KIND] = DBG$K_DISP_NORMAL;
;  1653     1772  3
;  1654     1773  3
;  1655     1774  3          ! Fill in the remaining contents of the new Screen Display Entry.
;  1656     1775  3          !
;  1657     1776  3          NEWDISP[DBG$B_DISP_REND] = .DISPTR[DBG$B_DISP_REND];
;  1658     1777  3          NEWDISP[DBG$W_DISP_FLAGS] = .DISPTR[DBG$W_DISP_FLAGS];
;  1659     1778  3          NEWDISP[DBG$W_DISP_RBEG] = .DISPTR[DBG$W_DISP_RBEG];
;  1660     1779  3          NEWDISP[DBG$W_DISP_RLEN] = .DISPTR[DBG$W_DISP_RLEN];
;  1661     1780  3          NEWDISP[DBG$W_DISP_CBEG] = .DISPTR[DBG$W_DISP_CBEG];
;  1662     1781  3          NEWDISP[DBG$W_DISP_CLEN] = .DISPTR[DBG$W_DISP_CLEN];
;  1663     1782  3          NEWDISP[DBG$W_DISP_DROW] = .DISPTR[DBG$W_DISP_DROW];
;  1664     1783  3          NEWDISP[DBG$W_DISP_DCOL] = .DISPTR[DBG$W_DISP_DCOL];
;  1665     1784  3          NEWDISP[DBG$W_DISP_MAX_LINECNT] = .DISPTR[DBG$W_DISP_MAX_LINECNT];
;  1666     1785  3          NEWDISP[DBG$L_DISP_START_LINE_PTR] = NEWDISP[DBG$L_DISP_START_LINE_PTR];
;  1667     1786  3          NEWDISP[DBG$L_DISP_END_LINE_PTR] = NEWDISP[DBG$L_DISP_START_LINE_PTR];
;  1668     1787  3          NEWDISP[DBG$L_DISP_MODPTR] = .DISPTR[DBG$L_DISP_MODPTR];
;  1669     1788  3          NEWDISP[DBG$L_DISP_CENTER] = .DISPTR[DBG$L_DISP_CENTER];
;  1670     1789  3          NEWDISP[DBG$L_DISP_MARKLINE] = .DISPTR[DBG$L_DISP_MARKLINE];
;  1671     1790  3          NEWDISP[DBG$L_DISP_MINLINE] = .DISPTR[DBG$L_DISP_MINLINE];
;  1672     1791  3          NEWDISP[DBG$L_DISP_MAXLINE] = .DISPTR[DBG$L_DISP_MAXLINE];
;  1673     1792  3
;  1674     1793  3
;  1675     1794  3          ! Mark the new display as being removed and invalidate its scrolling
;  1676     1795  3          ! count.
;  1677     1796  3          !
;  1678     1797  3          NEWDISP[DBG$V_DISP_REMOVE] = TRUE;
;  1679     1798  3          NEWDISP[DBG$V_DISP_INVSCR] = TRUE;
```

```
;  1680          1799  3
;  1681          1800  3
;  1682          1801  3          ! Loop over the old display's Display Line Entries to make copies of
;  1683          1802  3          ! all such lines in the new display.
;  1684          1803  3          !
;  1685          1804  3          DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  1686          1805  3          WHILE .DLEPTR NEQ DISPTR[DBG$L_DISP_START_LINE_PTR] DO
;  1687          1806  4              BEGIN
;  1688          1807  4
;  1689          1808  4
;  1690          1809  4              ! If this is a Source Display Line Entry, allocate another such
;  1691          1810  4              ! entry and fill in the fields specific to this kind of line entry.
;  1692          1811  4              !
;  1693          1812  4              IF .DLEPTR[DBG$V_DLINE_SOURCEFLG]
;  1694          1813  4              THEN
;  1695          1814  5                  BEGIN
;  1696          1815  5                  TEXTPTR = DLEPTR[DBG$A_DLINE_TEXT2];
;  1697          1816  5                  NEWDLE = DBG$GET_MEMORY(DBG$K_DLINE_ENTSIZE2
;  1698          1817  5                                          + .TEXTPTR[0]/%UPVAL + 1);
;  1699          1818  5                  NEWDLE[DBG$B_DLINE_REND] = .DLEPTR[DBG$B_DLINE_REND];
;  1700          1819  5                  NEWDLE[DBG$V_DLINE_SOURCEFLG] = TRUE;
;  1701          1820  5                  NEWDLE[DBG$W_DLINE_FILEID] = .DLEPTR[DBG$W_DLINE_FILEID];
;  1702          1821  5                  NEWDLE[DBG$L_DLINE_RECNUM] = .DLEPTR[DBG$L_DLINE_RECNUM];
;  1703          1822  5                  NEWDLE[DBG$L_DLINE_LINUM] = .DLEPTR[DBG$L_DLINE_LINUM];
;  1704          1823  5                  CH$MOVE(.TEXTPTR[0] + 1, TEXTPTR[0], NEWDLE[DBG$A_DLINE_TEXT2]);
;  1705          1824  5                  END
;  1706          1825  5
;  1707          1826  5
;  1708          1827  5              ! And for any other kind of display, we allocate a normal Display
;  1709          1828  5              ! Line Entry and fill in the line's text.  Note that we include
;  1710          1829  5              ! the rendition vector at the end of the text if there is one.
;  1711          1830  5              !
;  1712          1831  4              ELSE
;  1713          1832  5                  BEGIN
;  1714          1833  5                  TEXTPTR = DLEPTR[DBG$A_DLINE_TEXT];
;  1715          1834  5                  TEXTCNT = .TEXTPTR[0];
;  1716          1835  5                  IF .DLEPTR[DBG$V_DLINE_RENDFLG] THEN TEXTCNT = 2*.TEXTCNT;
;  1717          1836  5                  NEWDLE = DBG$GET_MEMORY(DBG$K_DLINE_ENTSIZE + .TEXTCNT/%UPVAL + 1);
;  1718          1837  5                  NEWDLE[DBG$B_DLINE_REND] = .DLEPTR[DBG$B_DLINE_REND];
;  1719          1838  5                  NEWDLE[DBG$V_DLINE_RENDFLG] = .DLEPTR[DBG$V_DLINE_RENDFLG];
;  1720          1839  5                  NEWDLE[DBG$B_DLINE_LENGTH] = .TEXTPTR[0];
;  1721          1840  5                  CH$MOVE(.TEXTCNT + 1, TEXTPTR[0], NEWDLE[DBG$A_DLINE_TEXT]);
;  1722          1841  4                  END;
;  1723          1842  4
;  1724          1843  4
;  1725          1844  4
;  1726          1845  4              ! If this is the first line in the display's screen window, set
;  1727          1846  4              ! the window pointer accordingly.
;  1728          1847  4              !
;  1729          1848  4              IF .DLEPTR EQL .DISPTR[DBG$L_DISP_WINDOW_PTR]
;  1730          1849  4              THEN
;  1731          1850  4                  NEWDISP[DBG$L_DISP_WINDOW_PTR] = .NEWDLE;
;  1732          1851  4
;  1733          1852  4
;  1734          1853  4              ! Link the new Display Line Entry into the Display Line Entry List
;  1735          1854  4              ! for the new display.  Also increment the display's line count.
;  1736          1855  4              !
```

```
;  1737          1856  4              DL_BLINK = .NEWDISP[DBG$L_DISP_END_LINE_PTR];
;  1738          1857  4              DL_FLINK = .DL_BLINK[DBG$L_DLINE_FLINK];
;  1739          1858  4              NEWDLE[DBG$L_DLINE_FLINK] = .DL_FLINK;
;  1740          1859  4              NEWDLE[DBG$L_DLINE_BLINK] = .DL_BLINK;
;  1741          1860  4              DL_FLINK[DBG$L_DLINE_BLINK] = .NEWDLE;
;  1742          1861  4              DL_BLINK[DBG$L_DLINE_FLINK] = .NEWDLE;
;  1743          1862  4              NEWDISP[DBG$W_DISP_LINECNT] = .NEWDISP[DBG$W_DISP_LINECNT] + 1;
;  1744          1863  4
;  1745          1864  4
;  1746          1865  4              ! Loop for the next Display Line Entry in the display.
;  1747          1866  4              !
;  1748          1867  4              DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  1749          1868  4
;  1750          1869  3              END;                            ! End of Display Line Entry loop
;  1751          1870  3
;  1752          1871  3
;  1753          1872  3          ! Link the new, fully built Screen Display Entry onto the Screen
;  1754          1873  3          ! Display Entry List.
;  1755          1874  3          !
;  1756          1875  3          DIS_FLINK = DBG$SCR_DISPLAY_LIST;
;  1757          1876  3          DIS_BLINK = .DIS_FLINK[DBG$L_DISP_BLINK];                                    ;
;  1758          1877  3          NEWDISP[DBG$L_DISP_FLINK] = .DIS_FLINK;
;  1759          1878  3          NEWDISP[DBG$L_DISP_BLINK] = .DIS_BLINK;
;  1760          1879  3          DIS_FLINK[DBG$L_DISP_BLINK] = .NEWDISP;
;  1761          1880  3          DIS_BLINK[DBG$L_DISP_FLINK] = .NEWDISP;
;  1762          1881  3
;  1763          1882  3
;  1764          1883  3          ! The current SAVE specification is fully processed.  Link to the next
;  1765          1884  3          ! Noun Node and loop.
;  1766          1885  3          !
;  1767          1886  3          NOUN_NODE = .NOUN_NODE[DBG$L_NOUN_LINK];
;  1768          1887  3
;  1769          1888  2              END;                                ! End of loop over Noun Nodes
;  1770          1889  2
;  1771          1890  2
;  1772          1891  2      ! All SAVE specifications have been executed.  Now return.
;  1773          1892  2      !
;  1774          1893  2      RETURN;
;  1775          1894  2
;  1776          1895  1      END;


                                        0FFC 00000          .ENTRY  DBG$SCR_EXECUTE_SAVE_CMD, Save R2,R3,R4,R5,-  ; 1685
                                                                    R6,R7,R8,R9,R10,R11                               :
                        5E                1C  C2 00002          SUBL2   #28, SP                                       :
                                  18      AE  D4 00005          CLRL    HANDLER_DISPTR                                ; 1706
                        6D      01C6      CF  DE 00008          MOVAL   13$, (FP)                                     :
                        50        04      AC  D0 0000D          MOVL    VERB_NODE, R0                                 ; 1735
                    14  AE        08      A0  D0 00011          MOVL    8(R0), NOUN_NODE                              :
                                          01  12 00016 1$:      BNEQ    2$                                            ; 1736
                                              04 00018          RET                                                   :
                52      14  AE            0C  C1 00019 2$:      ADDL3   #12, NOUN_NODE, R2                            ; 1744
                                          62  DD 0001E          PUSHL   (R2)                                          :
                    0000V   CF            01  FB 00020          CALLS   #1, DBG$SCR_LOOKUP_DISPLAY                    :
```

```
                        5B              50  D0 00025          MOVL    R0, DISPTR                                      ;
                                        16  13 00028          BEQL    3$                                              ; 1745
                52      14      AE      0C  C1 0002A          ADDL3   #12, NOUN_NODE, R2                              ; 1747
                                        62  DD 0002F          PUSHL   (R2)                                            ;
                                        01  DD 00031          PUSHL   #1                                              ;
                            00028372    8F  DD 00033          PUSHL   #164722                                         ;
                    00000000G   00      03  FB 00039          CALLS   #3, LIB$SIGNAL                                  ;
                        5B          14  BE  D0 00040  3$:     MOVL    @NOUN_NODE, DISPTR                              ; 1753
                50      14      AE      0C  C1 00044          ADDL3   #12, NOUN_NODE, R0                              ; 1754
                        59              60  D0 00049          MOVL    (R0), NAMEPTR                                   ;
                                    18  AE  D4 0004C          CLRL    HANDLER_DISPTR                                  ; 1755
                        50              69  9A 0004F          MOVZBL  (NAMEPTR), R0                                   ; 1756
                        50              04  C6 00052          DIVL2   #4, R0                                          ;
                                    14  A0  9F 00055          PUSHAB  20(R0)                                          ;
                    00000000G   00      01  FB 00058          CALLS   #1, DBG$GET_MEMORY                              ;
                        5A              50  D0 0005F          MOVL    R0, NEWDISP                                     ;
                            18  AE      5A  D0 00062          MOVL    NEWDISP, HANDLER_DISPTR                         ; 1757
                        50              69  9A 00066          MOVZBL  (NAMEPTR), R0                                   ; 1765
                                        50  D6 00069          INCL    R0                                              ;
            4C  AA              69      50  28 0006B          MOVC3   R0, (NAMEPTR), 76(NEWDISP)                      ;
                        50          08  AA  9E 00070          MOVAB   8(NEWDISP), R0                                  ; 1768
                        03          08  AB  91 00074          CMPB    8(DISPTR), #3                                   ; 1766
                                        05  12 00078          BNEQ    4$                                              ;
                        60              03  90 0007A          MOVB    #3, (R0)                                        ; 1768
                                        03  11 0007D          BRB     5$                                              ;
                        60              01  90 0007F  4$:     MOVB    #1, (R0)                                        ; 1771
                    01  A0          09  AB  90 00082  5$:     MOVB    9(DISPTR), 1(R0)                                ; 1776
                    02  A0          0A  AB  B0 00087          MOVW    10(DISPTR), 2(R0)                               ; 1777
                    10  AA          10  AB  7D 0008C          MOVQ    16(DISPTR), 16(NEWDISP)                         ; 1778
                    18  AA          18  AB  D0 00091          MOVL    24(DISPTR), 24(NEWDISP)                         ; 1782
                    1C  AA          1C  AB  B0 00096          MOVW    28(DISPTR), 28(NEWDISP)                         ; 1784
                    20  AA          20  AA  9E 0009B          MOVAB   32(NEWDISP), 32(NEWDISP)                        ; 1785
                    24  AA          20  AA  9E 000A0          MOVAB   32(NEWDISP), 36(NEWDISP)                        ; 1786
                    34  AA          34  AB  7D 000A5          MOVQ    52(DISPTR), 52(NEWDISP)                         ; 1787
                    3C  AA          3C  AB  7D 000AA          MOVQ    60(DISPTR), 60(NEWDISP)                         ; 1789
                    44  AA          44  AB  D0 000AF          MOVL    68(DISPTR), 68(NEWDISP)                         ; 1791
                    02  A0              03  88 000B4          BISB2   #3, 2(R0)                                       ; 1798
                        56          20  AB  D0 000B8          MOVL    32(DISPTR), DLEPTR                              ; 1804
                        50          20  AB  9E 000BC  6$:     MOVAB   32(DISPTR), R0                                  ; 1805
                        50              56  D1 000C0          CMPL    DLEPTR, R0                                      ;
                                        03  12 000C3          BNEQ    7$                                              ;
                                    00D9  31 000C5          BRW     12$                                             ;
                            52      08  A6  9E 000C8  7$:     MOVAB   8(DLEPTR), R2                                   ; 1812
            4F                      62      09  E1 000CC          BBC     #9, (R2), 8$                                    ;
                            57      14  A6  9E 000D0          MOVAB   20(R6), TEXTPTR                                 ; 1815
                        50              67  9A 000D4          MOVZBL  (TEXTPTR), R0                                   ; 1817
                        50              04  C6 000D7          DIVL2   #4, R0                                          ;
                                    06  A0  9F 000DA          PUSHAB  6(R0)                                           ;
                    00000000G   00      01  FB 000DD          CALLS   #1, DBG$GET_MEMORY                              ;
                        10  AE          50  D0 000E4          MOVL    R0, NEWDLE                                      ;
            50          10  AE          08  C1 000E8          ADDL3   #8, NEWDLE, R0                                  ; 1818
                        60              62  90 000ED          MOVB    (R2), (R0)                                      ;
            50          10  AE          09  C1 000F0          ADDL3   #9, NEWDLE, R0                                  ; 1819
                        60              02  88 000F5          BISB2   #2, (R0)                                        ;
            50          10  AE          0A  C1 000F8          ADDL3   #10, NEWDLE, R0                                 ; 1820
                        60      02      A2  B0 000FD          MOVW    2(R2), (R0)                                     ;
            50          10  AE          0C  C1 00101          ADDL3   #12, NEWDLE, R0                                 ; 1821
```

```
                        60      0C   A6  D0 00106          MOVL    12(DLEPTR), (R0)                        ;
                50      10      AE       10 C1 0010A       ADDL3   #16, NEWDLE, R0                         ; 1822
                        60      10   A6  D0 0010F          MOVL    16(DLEPTR), (R0)                        ;
                        50           67  9A 00113          MOVZBL  (TEXTPTR), R0                           ; 1823
                                     50  D6 00116          INCL    R0                                      ;
                7E      10      AE       14 C1 00118       ADDL3   #20, NEWDLE, -(SP)                      ;
                                     44  11 0011D          BRB     10$                                     ;
                        57      0C   A6  9E 0011F 8$:      MOVAB   12(DLEPTR), TEXTPTR                     ; 1833
                        58           67  9A 00123          MOVZBL  (TEXTPTR), TEXTCNT                      ; 1834
                        03      01   A2  E9 00126          BLBC    1(R2), 9$                               ; 1835
                        58           02  C4 0012A          MULL2   #2, TEXTCNT                             ;
                50      58           04  C7 0012D 9$:      DIVL3   #4, TEXTCNT, R0                         ; 1836
                                04   A0  9F 00131          PUSHAB  4(R0)                                   ;
                    00000000G   00       01  FB 00134      CALLS   #1, DBG$GET_MEMORY                      ;
                        10      AE       50  D0 0013B      MOVL    R0, NEWDLE                              ;
                50      10      AE       08  C1 0013F      ADDL3   #8, NEWDLE, R0                          ; 1837
                        60               62  90 00144      MOVB    (R2), (R0)                              ;
                50      10      AE       09  C1 00147      ADDL3   #9, NEWDLE, R0                          ; 1838
        60      01              00   01  A2  F0 0014C      INSV    1(R2), #0, #1, (R0)                     ;
                50      10      AE       0A  C1 00152      ADDL3   #10, NEWDLE, R0                         ; 1839
                        60               67  90 00157      MOVB    (TEXTPTR), (R0)                         ;
                        50           01  A8  9E 0015A      MOVAB   1(R8), R0                               ; 1840
                7E      10      AE       0C  C1 0015E      ADDL3   #12, NEWDLE, -(SP)                      ;
                9E      67               50  28 00163 10$:  MOVC3   R0, (TEXTPTR), @(SP)+                  ;
                        28      AB       56  D1 00167      CMPL    DLEPTR, 40(DISPTR)                      ; 1848
                                         05  12 0016B      BNEQ    11$                                     ;
                        28      AA   10  AE  D0 0016D      MOVL    NEWDLE, 40(NEWDISP)                     ; 1850
                        04      AE   24  AA  D0 00172 11$:  MOVL    36(NEWDISP), DL_BLINK                   ; 1856
                        08      AE   04  BE  D0 00177      MOVL    @DL_BLINK, DL_FLINK                     ; 1857
                        10      BE   08  AE  D0 0017C      MOVL    DL_FLINK, @NEWDLE                       ; 1858
                50      10      AE       04  C1 00181      ADDL3   #4, NEWDLE, R0                          ; 1859
                        60           04  AE  D0 00186      MOVL    DL_BLINK, (R0)                          ;
                50      08      AE       04  C1 0018A      ADDL3   #4, DL_FLINK, R0                        ; 1860
                        60           10  AE  D0 0018F      MOVL    NEWDLE, (R0)                            ;
                        04      BE   10  AE  D0 00193      MOVL    NEWDLE, @DL_BLINK                       ; 1861
                                     1E  AA  B6 00198      INCW    30(NEWDISP)                             ; 1862
                        56               66  D0 0019B      MOVL    (DLEPTR), DLEPTR                        ; 1867
                                     FF1B    31 0019E      BRW     6$                                      ; 1805
                        0C   AE 00000000'    EF  9E 001A3  12$:    MOVAB   DBG$SCR_DISPLAY_LIST, DIS_FLINK ; 1875
                50      0C   AE             04  C1 001A9  ADDL3   #4, DIS_FLINK, R0                       ; 1876
                             6E              60  D0 001AE  MOVL    (R0), DIS_BLINK                         ;
                             6A          0C  AE  D0 001B1  MOVL    DIS_FLINK, (NEWDISP)                    ; 1877
                        04   AA              6E  D0 001B5  MOVL    DIS_BLINK, 4(NEWDISP)                   ; 1878
                50      0C   AE             04  C1 001B9  ADDL3   #4, DIS_FLINK, R0                       ; 1879
                             60              5A  D0 001BE  MOVL    NEWDISP, (R0)                           ;
                        00   BE              5A  D0 001C1  MOVL    NEWDISP, @DIS_BLINK                     ; 1880
                50      14   AE             08  C1 001C5  ADDL3   #8, NOUN_NODE, R0                       ; 1886
                        14   AE              60  D0 001CA  MOVL    (R0), NOUN_NODE                         ;
                                         FE45    31 001CE  BRW     1$                                      ; 1736
                                                 04 001D1  RET                                             ; 1895
                                             0000 001D2 13$:  .WORD   Save nothing                         ; 1706
                        50      08   AC  D0 001D4          MOVL    8(AP), R0                               ;
                        50      04   A0  D0 001D8          MOVL    4(R0), R0                               ;
                                FC   A0  9F 001DC          PUSHAB  HANDLER_DISPTR                          ;
                                     01  DD 001DF          PUSHL   #1                                      ;
                                     5E  DD 001E1          PUSHL   SP                                      ;
                        7E      04   AC  7D 001E3          MOVQ    4(AP), -(SP)                            ;
```

```
                    0000V  CF          03  FB 001E7          CALLS   #3, HANDLER_EXECUTE_SAVE
                                           04 001EC          RET

; Routine Size:  493 bytes,    Routine Base:  DBG$CODE + 1101
```

```
;  1778          1896  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_SCROLL_CMD(VERB_NODE): NOVALUE =
;  1779          1897  1 !
;  1780          1898  1 ! FUNCTION
;  1781          1899  1 !       This routine executes the SCROLL command.  It accepts the address of
;  1782          1900  1 !       the verb node for the command as input and then extracts the command
;  1783          1901  1 !       parameters and executes the command.  Most of the real work of scrol-
;  1784          1902  1 !       ling the specified display is actually done by DBG$SCR_SCROLL_DISPLAY
;  1785          1903  1 !       which is called from this routine.
;  1786          1904  1 !
;  1787          1905  1 ! INPUTS
;  1788          1906  1 !       VERB_NODE - A pointer to the verb node for the SCROLL command to be
;  1789          1907  1 !                   executed.  The verb node, along with its attached noun node,
;  1790          1908  1 !                   contains all information picked up during the parsing of the
;  1791          1909  1 !                   command.
;  1792          1910  1 !
;  1793          1911  1 ! OUTPUTS
;  1794          1912  1 !       NONE
;  1795          1913  1 !
;  1796          1914  1
;  1797          1915  2     BEGIN
;  1798          1916  2
;  1799          1917  2     MAP
;  1800          1918  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input verb node
;  1801          1919  2
;  1802          1920  2     LOCAL
;  1803          1921  2         AMOUNT,                         ! Amount to scroll specified display
;  1804          1922  2         DIRECTION,                      ! Direction to scroll the display
;  1805          1923  2         DISPID: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;  1806          1924  2         NOUN_NODE: REF DBG$NOUN_NODE;   ! Pointer to Noun Node with Display ID
;  1807          1925  2
;  1808          1926  2
;  1809          1927  2
;  1810          1928  2     ! Get the scrolling direction and amount and the Display ID of the display
;  1811          1929  2     ! to be scrolled.  These all come from the Noun Node.
;  1812          1930  2     !
;  1813          1931  2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
;  1814          1932  2     DISPID = .NOUN_NODE[DBG$L_NOUN_VALUE];
;  1815          1933  2     DIRECTION = .NOUN_NODE[DBG$L_NOUN_VALUE2];
;  1816          1934  2     AMOUNT = .NOUN_NODE[DBG$L_NOUN_VALUE3];
;  1817          1935  2
;  1818            `6  2
;  1819            .,7 2     ! Now scroll the selected display in the desired direction and in the
;  1820          1938  2     ! desired amount.  Then return.
;  1821          1939  2     !
;  1822          1940  2     DBG$SCR_SCROLL_DISPLAY(.DISPID, .DIRECTION, .AMOUNT);
;  1823          1941  2     RETURN;
;  1824          1942  2
;  1825          1943  1     END;


                                0004 00000          .ENTRY  DBG$SCR_EXECUTE_SCROLL_CMD, Save R2          ; 1896
                    50      04  AC   D0 00002       MOVL    VERB_NODE, R0                                ; 1931
                    50      08  A0   D0 00006       MOVL    8(R0), NOUN_NODE                             ;
                    52          60   D0 0000A       MOVL    (NOUN_NODE), DISPID                          ; 1932
```

```
                      51      0C    A0  DD 0000D        MOVL    12(NOUN_NODE), DIRECTION                ; 1933
                      50      10    A0  DD 00011        MOVL    16(NOUN_NODE), AMOUNT                   ; 1934
                                    50  DD 00015        PUSHL   AMOUNT                                  ; 1940
                                    51  DD 00017        PUSHL   DIRECTION                               ;
                                    52  DD 00019        PUSHL   DISPID                                  ;
                            0000V   CF      03  FB 0001B        CALLS   #3, DBG$SCR_SCROLL_DISPLAY      ;
                                            04  00020        RET                                     ; 1943

; Routine Size:  33 bytes,    Routine Base:  DBG$CODE + 12EE
```

```
;1827     1944  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_SELECT_CMD(VERB_NODE): NOVALUE =
;1828     1945  1 !
;1829     1946  1 ! FUNCTION
;1830     1947  1 !       This routine executes the SELECT command.  It accepts the address of
;1831     1948  1 !       a verb node for the command as input, extracts the command parameters
;1832     1949  1 !       from that node, and executes the semantics of the command.  The seman-
;1833     1950  1 !       tics is to set the appropriate own variables (DBG$SCR_CURDISP_xxx) and
;1834     1951  1 !       global variables (DBG$GL_SCREEN_xxx) to point to the selected display's
;1835     1952  1 !       Screen Display Entry.
;1836     1953  1 !
;1837     1954  1 ! INPUTS
;1838     1955  1 !       VERB_NODE - A pointer to the verb node for the SELECT command to be
;1839     1956  1 !                   executed.  The verb node along with its attached noun node
;1840     1957  1 !                   contains all information picked up during the parsing of
;1841     1958  1 !                   the command.
;1842     1959  1 !
;1843     1960  1 ! OUTPUTS
;1844     1961  1 !       NONE
;1845     1962  1 !
;1846     1963  1
;1847     1964  2     BEGIN
;1848     1965  2
;1849     1966  2     MAP
;1850     1967  2         VERB_NODE: REF DBG$VEPB_NODE;   ! Pointer to input Verb Node
;1851     1968  2
;1852     1969  2     LOCAL
;1853     1970  2         DISPID: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;1854     1971  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Pointer to Noun Node with parse info
;1855     1972  2         SELECT_BITS: BITVECTOR[32];     ! Selection bits for different kinds
;1856     1973  2
;1857     1974  2
;1858     1975  2
;1859     1976  2     ! Get the selection kind bits (for history, input, output, scroll, or
;1860     1977  2     ! source) and the Display ID of the selected display from the Noun Node.
;1861     1978  2     !
;1862     1979  2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
;1863     1980  2     SELECT_BITS = .NOUN_NODE[DBG$L_NOUN_VALUE];
;1864     1981  2     DISPID = .NOUN_NODE[DBG$L_NOUN_VALUE2];
;1865     1982  2
;1866     1983  2
;1867     1984  2     ! Select a new history output display.  Only displays of the NORMAL kind
;1868     1985  2     ! are allowed to be history displays.
;1869     1986  2     !
;1870     1987  2     IF .SELECT_BITS[0]
;1871     1988  2     THEN
;1872     1989  3         BEGIN
;1873     1990  3         IF .DISPID NEQ 0
;1874     1991  3         THEN
;1875     1992  4             BEGIN
;1876     1993  4             IF .DISPID[DBG$B_DISP_KIND] NEQ DBG$K_DISP_NORMAL
;1877     1994  4             THEN
;1878     1995  4                 SIGNAL(DBG$_INVSELDIS, 1, DISPID[DBG$A_DISP_NAME]);
;1879     1996  4
;1880     1997  3             END;
;1881     1998  3
;1882     1999  3         DBG$GL_SCREEN_HISTORY = .DISPID;
;1883     2000  2         END;
```

```
;  1884          2001  2
;  1885          2002  2
;  1886          2003  2      ! Select a new command input display.  Only displays of kind NORMAL or DO
;  1887          2004  2      ! are allowed to be input displays.
;  1888          2005  2      !
;  1889          2006  2      IF .SELECT_BITS[1]
;  1890          2007  2      THEN
;  1891          2008  3          BEGIN
;  1892          2009  3          IF .DISPID NEQ 0
;  1893          2010  3          THEN
;  1894          2011  4              BEGIN
;  1895          2012  4              IF (.DISPID[DBG$B_DISP_KIND] NEQ DBG$K_DISP_NORMAL) AND
;  1896          2013  5                 (.DISPID[DBG$B_DISP_KIND] NEQ DBG$K_DISP_DO)
;  1897          2014  4              THEN
;  1898          2015  4                  SIGNAL(DBG$_INVSELDIS, 1, DISPID[DBG$A_DISP_NAME]);
;  1899          2016  4
;  1900          2017  3              END;
;  1901          2018  3
;  1902          2019  3          DBG$SCR_CURDISP_INPUT = .DISPID;
;  1903          2020  3          IF .DISPID EQL 0 THEN DBG$GL_SCREEN_INPUT = 0;
;  1904          2021  3          IF .DBG$GL_SCREEN_MODE THEN DBG$GL_SCREEN_INPUT = .DISPID;
;  1905          2022  2          END;
;  1906          2023  2
;  1907          2024  2
;  1908          2025  2      ! Select a new command output display.  Only displays of kinds NORMAL or DO
;  1909          2026  2      ! are allowed to be output displays.
;  1910          2027  2      !
;  1911          2028  2      IF .SELECT_BITS[2]
;  1912          2029  2      THEN
;  1913          2030  3          BEGIN
;  1914          2031  3          IF .DISPID NEQ 0
;  1915          2032  3          THEN
;  1916          2033  4              BEGIN
;  1917          2034  4              IF (.DISPID[DBG$B_DISP_KIND] NEQ DBG$K_DISP_NORMAL) AND
;  1918          2035  5                 (.DISPID[DBG$B_DISP_KIND] NEQ DBG$K_DISP_DO)
;  1919          2036  4              THEN
;  1920          2037  4                  SIGNAL(DBG$_INVSELDIS, 1, DISPID[DBG$A_DISP_NAME]);
;  1921          2038  4
;  1922          2039  3              END;
;  1923          2040  3
;  1924          2041  3          DBG$SCR_CURDISP_OUTPUT = .DISPID;
;  1925          2042  3          IF .DISPID EQL 0 THEN DBG$GL_SCREEN_OUTPUT = 0;
;  1926          2043  3          IF .DBG$GL_SCREEN_MODE THEN DBG$GL_SCREEN_OUTPUT = .DISPID;
;  1927          2044  2          END;
;  1928          2045  2
;  1929          2046  2
;  1930          2047  2      ! Select a new default scrolling display for the SCROLL command.  Any kind
;  1931          2048  2      ! of display can be scrolled.
;  1932          2049  2      !
;  1933          2050  2      IF .SELECT_BITS[3]
;  1934          2051  2      THEN
;  1935          2052  2          DBG$SCR_CURDISP_SCROLL = .DISPID;
;  1936          2053  2
;  1937          2054  2
;  1938          2055  2      ! Select a new source output display.  Only displays with SOURCE content
;  1939          2056  2      ! specifications can be selected as source output displays.
;  1940          2057  2      !
```

```
;  1941          2058  2      IF .SELECT_BITS[4]
;  1942          2059  2      THEN
;  1943          2060  3          BEGIN
;  1944          2061  3          IF .DISPID NEQ 0
;  1945          2062  3          THEN
;  1946          2063  4              BEGIN
;  1947          2064  4              IF .DISPID[DBG$B_DISP_KIND] NEQ DBG$K_DISP_SOURCE
;  1948          2065  4              THEN
;  1949          2066  4                  SIGNAL(DBG$_INVSELDIS, 1, DISPID[DBG$A_DISP_NAME]);
;  1950          2067  4
;  1951          2068  3              END;
;  1952          2069  3
;  1953          2070  3          DBG$SCR_CURDISP_SOURCE = .DISPID;
;  1954          2071  3          IF .DISPID EQL 0 THEN DBG$GL_SCREEN_SOURCE = 0;
;  1955          2072  3          IF .DBG$GL_SCREEN_MODE THEN DBG$GL_SCREEN_SOURCE = .DISPID;
;  1956          2073  2          END;
;  1957          2074  2
;  1958          2075  2
;  1959          2076  2      ! The command processing is completed.  Now return.
;  1960          2077  2      !
;  1961          2078  2      RETURN;
;  1962          2079  2
;  1963          2080  1      END;


                                007C 00000          .ENTRY  DBG$SCR_EXECUTE_SELECT_CMD, Save R2,R3,R4,- ; 1944
                                                            R5,R6
                            56 00000000G   00  9E 00002     MOVAB   LIB$SIGNAL, R6
                            55 00000000'   EF  9E 00009     MOVAB   DBG$SCR_CURDISP_INPUT, R5
                            54 00000000'   EF  9E 00010     MOVAB   DBG$GL_SCREEN_MODE, R4
                            50       04    AC  D0 00017     MOVL    VERB_NODE, R0                       ; 1979
                            50       08    A0  D0 0001B     MOVL    8(R0), NOUN_NODE
                            53             60  D0 0001F     MOVL    (NOUN_NODE), SELECT_BITS            ; 1980
                            52       0C    A0  D0 00022     MOVL    12(NOUN_NODE), DISPID               ; 1981
                            1A             53  E9 00026     BLBC    SELECT_BITS, 2$                     ; 1987
                                           14  13 00029     BEQL    1$                                  ; 1990
                            01       08    A2  91 0002B     CMPB    8(DISPID), #1                       ; 1993
                                           0E  13 0002F     BEQL    1$
                                     4C    A2  9F 00031     PUSHAB  76(DISPID)                          ; 1995
                                           01  DD 00034     PUSHL   #1
                                00028A9A   8F  DD 00036     PUSHL   #166554
                            66             03  FB 0003C     CALLS   #3, LIB$SIGNAL
                      F4    A4             52  D0 0003F 1$: MOVL    DISPID, DBG$GL_SCREEN_HISTORY       ; 1999
                   28       53             01  E1 00043 2$: BBC     #1, SELECT_BITS, 5$                 ; 2006
                                           1A  13 00047     BEQL    3$                                  ; 2009
                            01       08    A2  91 00049     CMPB    8(DISPID), #1                       ; 2012
                                           14  13 0004D     BEQL    3$
                            02       08    A2  91 0004F     CMPB    8(DISPID), #2                       ; 2013
                                           0E  13 00053     BEQL    3$
                                     4C    A2  9F 00055     PUSHAB  76(DISPID)                          ; 2015
                                           01  DD 00058     PUSHL   #1
                                00028A9A   8F  DD 0005A     PUSHL   #166554
                            66             03  FB 00060     CALLS   #3, LIB$SIGNAL
                            65             52  D0 00063 3$: MOVL    DISPID, DBG$SCR_CURDISP_INPUT       ; 2019
```

```
                                03  12 00066          BNEQ    4$                                      ; 2020
                             F8 A4  D4 00068          CLRL    DBG$GL_SCREEN_INPUT
                          04    64  E9 0006B 4$:      BLBC    DBG$GL_SCREEN_MODE, 5$                  ; 2021
                       F8 A4    52  D0 0006E          MOVL    DISPID, DBG$GL_SCREEN_INPUT
                    2E 53       02  E1 00072 5$:      BBC     #2, SELECT_BITS, 8$                     ; 2028
                                52  D5 00076          TSTL    DISPID                                  ; 2031
                                1A  13 00078          BEQL    6$
                          01 08 A2  91 0007A          CMPB    8(DISPID), #1                           ; 2034
                                14  13 0007E          BEQL    6$
                          02 08 A2  91 00080          CMPB    8(DISPID), #2                           ; 2035
                                0E  13 00084          BEQL    6$
                             4C A2  9F 00086          PUSHAB  76(DISPID)                              ; 2037
                                01  DD 00089          PUSHL   #1
                   00028A9A     8F  DD 0008B          PUSHL   #166554
                          66    03  FB 00091          CALLS   #3, LIB$SIGNAL
                       04 A5    52  D0 00094 6$:      MOVL    DISPID, DBG$SCR_CURDISP_OUTPUT          ; 2041
                                03  12 00098          BNEQ    7$                                      ; 2042
                             08 A4  D4 0009A          CLRL    DBG$GL_SCREEN_OUTPUT
                          04    64  E9 0009D 7$:      BLBC    DBG$GL_SCREEN_MODE, 8$                  ; 2043
                       08 A4    52  D0 000A0          MOVL    DISPID, DBG$GL_SCREEN_OUTPUT
                    04 53       03  E1 000A4 8$:      BBC     #3, SELECT_BITS, 9$                     ; 2050
                       08 A5    52  D0 000A8          MOVL    DISPID, DBG$SCR_CURDISP_SCROLL          ; 2052
                    28 53       04  E1 000AC 9$:      BBC     #4, SELECT_BITS, 12$                    ; 2058
                                52  D5 000B0          TSTL    DISPID                                  ; 2061
                                14  13 000B2          BEQL    10$
                          03 08 A2  91 000B4          CMPB    8(DISPID), #3                           ; 2064
                                0E  13 000B8          BEQL    10$
                             4C A2  9F 000BA          PUSHAB  76(DISPID)                              ; 2066
                                01  DD 000BD          PUSHL   #1
                   00028A9A     8F  DD 000BF          PUSHL   #166554
                          66    03  FB 000C5          CALLS   #3, LIB$SIGNAL
                       0C A5    52  D0 000C8 10$:     MOVL    DISPID, DBG$SCR_CURDISP_SOURCE          ; 2070
                                03  12 000CC          BNEQ    11$                                     ; 2071
                             0C A4  D4 000CE          CLRL    DBG$GL_SCREEN_SOURCE
                          04    64  E9 000D1 11$:     BLBC    DBG$GL_SCREEN_MODE, 12$                 ; 2072
                       0C A4    52  D0 000D4          MOVL    DISPID, DBG$GL_SCREEN_SOURCE
                                04  000D8 12$:      RET                                             ; 2080

; Routine Size:  217 bytes,    Routine Base:  DBG$CODE + 130F
```

```
;  1965          2081  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_SETTERM_CMD(VERB_NODE): NOVALUE =
;  1966          2082  1 !
;  1967          2083  1 ! FUNCTION
;  1968          2084  1 !       This routine executes the SET TERMINAL command.  The only information
;  1969          2085  1 !       the SET TERMINAL command can impart at present is the terminal width
;  1970          2086  1 !       via the /WIDTH:n qualifier.  This routine thus picks up the new termi-
;  1971          2087  1 !       nal width, stores it in the global variable DBG$SRC_TERM_WIDTH, and
;  1972          2088  1 !       invalidates all screen display lines currently on the screen to force
;  1973          2089  1 !       those lines to be rewritten using the new line width.  No attempt is
;  1974          2090  1 !       made to reset the terminal with through any escape sequence, however.
;  1975          2091  1 !       The global variable DBG$SRC_TERM_WIDTH controls the folding of all
;  1976          2092  1 !       print and source lines and the truncation of screen display output.
;  1977          2093  1 !
;  1978          2094  1 ! INPUTS
;  1979          2095  1 !       VERB_NODE - A pointer to the Verb Node for the SET TERMINAL command
;  1980          2096  1 !                   to be executed.  This Verb Node and the attached Noun Node
;  1981          2097  1 !                   contain all information picked up during the parsing of
;  1982          2098  1 !                   the command.
;  1983          2099  1 !
;  1984          2100  1 ! OUTPUTS
;  1985          2101  1 !       NONE
;  1986          2102  1
;  1987          2103  2     BEGIN
;  1988          2104  2
;  1989          2105  2     MAP
;  1990          2106  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  1991          2107  2
;  1992          2108  2     LOCAL
;  1993          2109  2         NOUN_NODE: REF DBG$NOUN_NODE;   ! Pointer to input Noun Node
;  1994          2110  2
;  1995          2111  2
;  1996          2112  2
;  1997          2113  2     ! Extract the new terminal with from the input Noun Node.  If we are in
;  1998          2114  2     ! Screen Mode, we also invalidate the current screen contents so the
;  1999          2115  2     ! screen gets refreshed.  We then return.
;  2000          2116  2     !
;  2001          2117  2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
;  2002          2118  2     DBG$SRC_TERM_WIDTH = .NOUN_NODE[DBG$L_NOUN_VALUE];
;  2003          2119  2     IF .DBG$GL_SCREEN_MODE
;  2004          2120  2     THEN
;  2005          2121  3         BEGIN
;  2006          2122  3         INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  2007          2123  3             OLD_VALID[.I] = FALSE;
;  2008          2124  3
;  2009          2125  2         END;
;  2010          2126  2
;  2011          2127  2     RETURN;
;  2012          2128  2
;  2013          2129  1     END;

                                  0000 00000          .ENTRY  DBG$SCR_EXECUTE_SETTERM_CMD, Save nothing   ; 2081
                        50    04   AC   D0 00002      MOVL    VERB_NODE, R0                               ; 2117
                        50    08   A0   D0 00006      MOVL    8(R0), NOUN_NODE                            :
```

```
                 00000000G  00              60  D0 0000A          MOVL    (NOUN_NODE), DBG$SRC_TERM_WIDTH          ; 2118
                            0E 00000000'    EF  E9 00011          BLBC    DBG$GL_SCREEN_MODE, 3$                   ; 2119
                                            50  D4 00018          CLRL    I                                        ; 2122
                00 00000000'  EF            50  E5 0001A 1$:      BBCC    I, OLD_VALID, 2$                         ; 2123
                F4            50            14  F3 00022 2$:      AOBLEQ  #20, I, 1$                               ;
                                                04 00026 3$:      RET                                              ; 2129

; Routine Size:  39 bytes,    Routine Base:  DBG$CODE + 13E8
```

```
;2015   2130  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_SETWIND_CMD(VERB_NODE): NOVALUE =
;2016   2131  1 !
;2017   2132  1 ! FUNCTION
;2018   2133  1 !       This routine executes the SET WINDOW command.  It accepts the address
;2019   2134  1 !       of the Verb Node for the command as input and then extracts the command
;2020   2135  1 !       parameters and executes the command.  The command parameters consist of
;2021   2136  1 !       a window name and a window specification, all encoded in a Noun Node.
;2022   2137  1 !
;2023   2138  1 ! INPUTS
;2024   2139  1 !       VERB_NODE - A pointer to the Verb Node for the SET WINDOW command to be
;2025   2140  1 !                   executed.  The Verb Node along with its attached Noun Node
;2026   2141  1 !                   contains all information picked up during the parsing of the
;2027   2142  1 !                   command.
;2028   2143  1 !
;2029   2144  1 ! OUTPUTS
;2030   2145  1 !       NONE                                                                    66
;2031   2146  1 !
;2032   2147  1
;2033   2148  2     BEGIN                                                                       21
;2034   2149  2
;2035   2150  2     MAP
;2036   2151  2         VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to input Verb Node
;2037   2152  2
;2038   2153  2     LOCAL
;2039   2154  2         CBEG,                            ! Beginning column location of window
;2040   2155  2         CLEN,                            ! Column length (width) of window
;2041   2156  2         COL_INFO,                        ! Encoded column parameters
;2042   2157  2         NAMEPTR: REF VECTOR[,BYTE],      ! Pointer to ASCIC window name
;2043   2158  2         NOUN_NODE: REF DBG$NOUN_NODE,    ! Pointer to current Noun Node
;2044   2159  2         RBEG,                            ! Beginning row location of window
;2045   2160  2         RLEN,                            ! Row length (height) of window
;2046   2161  2         ROW_INFO;                        ! Encoded row parameters
;2047   2162  2
;2048   2163  2
;2049   2164  2
;2050   2165  2     ! Extract all the window parameters from the Noun Node.
;2051   2166  2     !
;2052   2167  2     NOUN_NODE = .VERB_NODE[DBG$L_VERB_OBJECT_PTR];
;2053   2168  2     NAMEPTR = .NOUN_NODE[DBG$L_NOUN_VALUE];
;2054   2169  2     ROW_INFO = .NOUN_NODE[DBG$L_NOUN_VALUE2];
;2055   2170  2     RBEG = .ROW_INFO<W0_>;
;2056   2171  2     RLEN = .ROW_INFO<W1_>;
;2057   2172  2     COL_INFO = .NOUN_NODE[DBG$L_NOUN_VALUE3];
;2058   2173  2     CBEG = .COL_INFO<W0_>;
;2059   2174  2     CLEN = .COL_INFO<W1_>;
;2060   2175  2
;2061   2176  2
;2062   2177  2     ! Now allocate and build a Screen Window Entry for the new screen window.
;2063   2178  2     ! Then return.
;2064   2179  2     !
;2065   2180  2     DBG$SCR_CREATE_WINDOW(.NAMEPTR, .RBEG, .RLEN, .CBEG, .CLEN);
;2066   2181  2     RETURN;
;2067   2182  2
;2068   2183  1     END;
```

```
                                                001C 00000        .ENTRY  DBG$SCR_EXECUTE_SETWIND_CMD, Save R2,R3,R4 ; 2130
                                     50    04   AC   D0 00002    MOVL    VERB_NODE, R0                              ; 2167
                                     50    08   A0   D0 00006    MOVL    8(R0), NOUN_NODE                           ;
                                     54         60   D0 0000A    MOVL    (NOUN_NODE), NAMEPTR                       ; 2168
                                     51    0C   A0   D0 0000D    MOVL    12(NOUN_NODE), ROW_INFO                    ; 2169
                                     53         51   3C 00011    MOVZWL  ROW_INFO, RBEG                             ; 2170
                 52        51                10   10   EF 00014    EXTZV   #16, #16, ROW_INFO, RLEN                   ; 2171
                                     50    10   A0   D0 00019    MOVL    16(NOUN_NODE), COL_INFO                    ; 2172
                                     51         50   3C 0001D    MOVZWL  COL_INFO, CBEG                             ; 2173
                 50        50                10   10   EF 00020    EXTZV   #16, #16, COL_INFO, CLEN                   ; 2174
                                               50   DD 00025    PUSHL   CLEN                                       ; 2180
                                               51   DD 00027    PUSHL   CBEG                                       ;
                                               52   DD 00029    PUSHL   RLEN                                       ;
                                               53   DD 0002B    PUSHL   RBEG                                       ;
                                               54   DD 0002D    PUSHL   NAMEPTR                                    ;
                          F713   CF            05   FB 0002F    CALLS   #5, DBG$SCR_CREATE_WINDOW                  ;
                                                    04 00034    RET                                                ; 2183

; Routine Size:  53 bytes,    Routine Base:  DBG$CODE + 140F
```

```
;  2070          2184  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_SHODISP_CMD(VERB_NODE): NOVALUE =
;  2071          2185  1 !
;  2072          2186  1 ! FUNCTION
;  2073          2187  1 !       This routine executes the SHOW DISPLAY command.  It accepts the
;  2074          2188  1 !       address of the verb node for the command as input and then executes
;  2075          2189  1 !       the command.  Since there are no command parameters, the Verb Node
;  2076          2190  1 !       is actually ignored.  The command processing consists of going through
;  2077          2191  1 !       all displays on the Screen Display List and listing out the name and
;  2078          2192  1 !       attributes of each such display.
;  2079          2193  1 !
;  2080          2194  1 ! INPUTS
;  2081          2195  1 !       VERB_NODE - A pointer to the Verb Node for the SHOW DISPLAY command to
;  2082          2196  1 !                   be executed.  The Verb Node is actually ignored.
;  2083          2197  1 !
;  2084          2198  1 ! OUTPUTS
;  2085          2199  1 !       NONE
;  2086          2200  1 !
;  2087          2201  1
;  2088          2202  2     BEGIN
;  2089          2203  2
;  2090          2204  2     LOCAL
;  2091          2205  2         DISPTR: REF DBG$DISP_ENTRY,      ! Pointer to current Display Entry
;  2092          2206  2         WPTR: REF DBG$WINDOW_ENTRY;      ! Pointer to a Screen Window Entry
;  2093          2207  2
;  2094          2208  2
;  2095          2209  2
;  2096          2210  2     ! If the Screen Display List is empty, print a message to that effect and
;  2097          2211  2     ! return immediately.
;  2098          2212  2     !
;  2099          2213  2     IF .DBG$SCR_DISPLAY_LIST[0] EQL DBG$SCR_DISPLAY_LIST
;  2100          2214  2     THEN
;  2101          2215  3         BEGIN
;  2102          2216  3         DBG$PRINT(UPLIT BYTE(%ASCIC 'no displays defined'), 0);
;  2103          2217  3         DBG$NEWLINE();
;  2104          2218  3         RETURN;
;  2105          2219  2         END;
;  2106          2220  2
;  2107          2221  2
;  2108          2222  2     ! Loop through all displays in the Screen Display List and print out the
;  2109          2223  2     ! name and attributes of each such display.
;  2110          2224  2     !
;  2111          2225  2     DISPTR = .DBG$SCR_DISPLAY_LIST[0];
;  2112          2226  2     WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;  2113          2227  3         BEGIN
;  2114          2228  3
;  2115          2229  3
;  2116          2230  3         ! Start by printing out the current display's name.
;  2117          2231  3         !
;  2118          2232  3         DBG$PRINT(UPLIT BYTE(%ASCIC 'display '''AC''), DISPTR[DBG$A_DISP_NAME]);
;  2119          2233  3
;  2120          2234  3
;  2121          2235  3         ! Next print out the window name.  Here we search the Screen Window
;  2122          2236  3         ! List to see if there is a screen window with the same window param-
;  2123          2237  3         ! eters as those of the current display.  If so, we print out that
;  2124          2238  3         ! window's name.
;  2125          2239  3         !
;  2126          2240  3         WPTR = .DBG$SCR_WINDOW_LIST[0];
```

```
;  2127     2241  3          WHILE .WPTR NEQ DBG$SCR_WINDOW_LIST DO
;  2128     2242  4              BEGIN
;  2129     2243  4              IF (.WPTR[DBG$W_WINDOW_RBEG] EQL .DISPTR[DBG$W_DISP_RBEG]) AND
;  2130     2244  4                 (.WPTR[DBG$W_WINDOW_RLEN] EQL .DISPTR[DBG$W_DISP_RLEN]) AND
;  2131     2245  4                 (.WPTR[DBG$W_WINDOW_CBEG] EQL .DISPTR[DBG$W_DISP_CBEG]) AND
;  2132     2246  5                 (.WPTR[DBG$W_WINDOW_CLEN] EQL .DISPTR[DBG$W_DISP_CLEN])
;  2133     2247  4              THEN
;  2134     2248  5                  BEGIN
;  2135     2249  5                  DBG$PRINT(UPLIT BYTE(%ASCIC ' at !AC'), WPTR[DBG$A_WINDOW_NAME]);
;  2136     2250  5                  WPTR = 0;
;  2137     2251  5                  EXITLOOP;
;  2138     2252  4                  END;
;  2139     2253  4
;  2140     2254  4              WPTR = .WPTR[DBG$L_WINDOW_FLINK];
;  2141     2255  3              END;
;  2142     2256  3
;  2143     2257  3
;  2144     2258  3          ! If no window with the same window parameters could be found, print
;  2145     2259  3          ! out the window parameters as a string of numbers.  This situation
;  2146     2260  3          ! normally only occurs if the user cancelled the window definition
;  2147     2261  3          ! which was used to define the original display position.
;  2148     2262  3          !
;  2149     2263  3          IF .WPTR NEQ 0
;  2150     2264  3          THEN
;  2151     2265  4              BEGIN                                   !<<-------- TEMPORARY ----------
;  2152     2266  5              IF .DBG$GL_DEVELOPER[7] THEN BEGIN      !<<-------- TEMPORARY ----------
;  2153     2267  5              DBG$PRINT(UPLIT BYTE(%ASCIC ' at (!SL,!SL,!SL,!SL)'),
;  2154     2268  5                        .DISPTR[DBG$W_DISP_RBEG], .DISPTR[DBG$W_DISP_RLEN],
;  2155     2269  5                        .DISPTR[DBG$W_DISP_CBEG], .DISPTR[DBG$W_DISP_CLEN]);
;  2156     2270  5              END ELSE BEGIN                          !<<-------- TEMPORARY ----------
;  2157     2271  5              DBG$PRINT(UPLIT BYTE(%ASCIC ' at (!SL,!SL)'),       !<<-------------
;  2158     2272  5                        .DISPTR[DBG$W_DISP_RBEG], .DISPTR[DBG$W_DISP_RLEN]);!<--
;  2159     2273  4              END;                                    !<<-------- TEMPORARY ----------
;  2160     2274  3              END;                                    !<<-------- TEMPORARY ----------
;  2161     2275  3
;  2162     2276  3
;  2163     2277  3          ! Print out the maximum size of the display in lines.
;  2164     2278  3          !
;  2165     2279  3          DBG$PRINT(UPLIT BYTE(%ASCIC ', size = !SL'),
;  2166     2280  3                    .DISPTR[DBG$W_DISP_MAX_LINE(NT]);
;  2167     2281  3
;  2168     2282  3
;  2169     2283  3          ! Print out the values of the removed and mark-changes flags if set.
;  2170     2284  3          !
;  2171     2285  3          IF .DISPTR[DBG$V_DISP_REMOVE]
;  2172     2286  3          THEN
;  2173     2287  3              DBG$PRINT(UPLIT BYTE(%ASCIC ', removed'), 0);
;  2174     2288  3
;  2175     2289  3          IF .DISPTR[DBG$V_DISP_MARKFLG]
;  2176     2290  3          THEN
;  2177     2291  3              DBG$PRINT(UPLIT BYTE(%ASCIC ', mark changes'), 0);
;  2178     2292  3
;  2179     2293  3
;  2180     2294  3          ! Print out the display's kind and the associated DEBUG command list
;  2181     2295  3          ! (if any).
;  2182     2296  3          !
;  2183     2297  3          CASE .DISPTR[DBG$B_DISP_KIND] FROM DBG$K_DISP_MINKIND
```

```
;  2184          2298  3                    TO DBG$K_DISP_MAXKIND OF
;  2185          2299  3            SET
;  2186          2300  3
;  2187          2301  3
;  2188          2302  3            ! Handle the NORMAL kind.  Here there is no DEBUG command list so
;  2189          2303  3            ! we just print "kind = NORMAL" on the same line.
;  2190          2304  3            !
;  2191          2305  3            [DBG$K_DISP_NORMAL]:
;  2192          2306  3                DBG$PRINT(UPLIT BYTE(%ASCIC ', kind = NORMAL'), 0);
;  2193          2307  3
;  2194          2308  3
;  2195          2309  3            ! Handle the DO kind.  Here we print out the DO command list.
;  2196          2310  3            ! The command list is printed on a separate line.
;  2197          2311  3            !
;  2198          2312  3            [DBG$K_DISP_DO]:
;  2199          2313  4                BEGIN
;  2200          2314  4                LOCAL
;  2201          2315  4                    CMDPTR: REF VECTOR[,WORD];
;  2202          2316  4                CMDPTR = .DISPTR[DBG$L_DISP_CMDLIST];
;  2203          2317  4                DBG$NEWLINE();
;  2204          2318  4                DBG$PRINT(UPLIT BYTE(%ASCIC '    kind = DO (!AD)'),
;  2205          2319  4                    .CMDPTR[0]-1, CMDPTR[1]);
;  2206          2320  3                END;
;  2207          2321  3
;  2208          2322  3
;  2209          2323  3            ! Handle the SOURCE kind.  Here we print out the SOURCE command
;  2210          2324  3            ! list if there is one.  The command list is always printed on
;  2211          2325  3            ! a separate line.
;  2212          2326  3            !
;  2213          2327  3            [DBG$K_DISP_SOURCE]:
;  2214          2328  4                BEGIN
;  2215          2329  4                IF .DISPTR[DBG$L_DISP_CMDLIST] EQL 0
;  2216          2330  4                THEN
;  2217          2331  4                    DBG$PRINT(UPLIT BYTE(%ASCIC ', kind = SOURCE'), 0)
;  2218          2332  4
;  2219          2333  4                ELSE
;  2220          2334  5                    BEGIN
;  2221          2335  5                    LOCAL
;  2222          2336  5                        CMDPTR: REF VECTOR[,WORD];
;  2223          2337  5                    CMDPTR = .DISPTR[DBG$L_DISP_CMDLIST];
;  2224          2338  5                    DBG$NEWLINE();
;  2225          2339  5                    DBG$PRINT(UPLIT BYTE(%ASCIC '    kind = SOURCE (!AD)'),
;  2226          2340  5                        .CMDPTR[0]-1, CMDPTR[1]);
;  2227          2341  4                    END;
;  2228          2342  4
;  2229          2343  3                END;
;  2230          2344  3
;  2231          2345  3
;  2232          2346  3            ! Handle the REGISTER display kind.  Simply print the kind on the
;  2233          2347  3            ! same line.
;  2234          2348  3            !
;  2235          2349  3            [DBG$K_DISP_REGISTER]:
;  2236          2350  3                DBG$PRINT(UPLIT BYTE(%ASCIC ', kind = REGISTER'), 0);
;  2237          2351  3
;  2238          2352  3
;  2239          2353  3            ! Any other value is an internal DEBUG error.
;  2240          2354  3            !
```

```
;  2241          2355  3          [INRANGE, OUTRANGE]:
;  2242          2356  3              $DBG_ERROR('DBGSCREEN\EXECUTE_SHODISP_CMD');
;  2243          2357  3
;  2244          2358  3          TES;
;  2245          2359  3
;  2246          2360  3
;  2247          2361  3      ! Flush out this print line and loop for the next display.
;  2248          2362  3      !
;  2249          2363  3      DBG$NEWLINE();
;  2250          2364  3      DISPTR = .DISPTR[DBG$L_DISP_FLINK];
;  2251          2365  3
;  2252          2366  2      END;                            ! End of loop over Display List
;  2253          2367  2
;  2254          2368  2
;  2255          2369  2  ! We are all done.  Return to the caller.
;  2256          2370  2  !
;  2257          2371  2  RETURN;
;  2258          2372  2
;  2259          2373  1  END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

65 64 20 73 79 61 6C 70 73 69 64 20 6F 6E 13 003FF P.AFB:  .ASCII  <19>\no displays defined\
                                  64 65 6E 69 66 0040E
         43 41 21 20 79 61 6C 70 73 69 64 0B 00413 P.AFC:  .ASCII  <11>\display !AC\
                     43 41 21 20 74 61 20 07 0041F P.AFD:  .ASCII  <7>\ at !AC\
21 2C 4C 53 21 2C 4C 53 21 28 20 74 61 20 15 00427 P.AFE:  .ASCII  <21>\ at (!SL,!SL,!SL,!SL)\
                        29 4C 53 21 2C 4C 53 00436
   29 4C 53 21 2C 4C 53 21 28 20 74 61 20 0D 0043D P.AFF:  .ASCII  <13>\ at (!SL,!SL)\
      4C 53 21 20 3D 20 65 7A 69 73 20 2C 0C 0044B P.AFG:  .ASCII  <12>\, size = !SL\
               64 65 76 6F 6D 65 72 20 2C 09 00458 P.AFH:  .ASCII  <9>\, removed\
73 65 67 6E 61 68 63 20 6B 72 61 6D 20 2C 0E 00462 P.AFI:  .ASCII  <14>\, mark changes\
41 4D 52 4F 4E 20 3D 20 64 6E 69 6B 20 2C 0F 00471 P.AFJ:  .ASCII  <15>\, kind = NORMAL\
                                                  4C 00480
20 4F 44 20 3D 20 64 6E 69 6B 20 20 20 20 13 00481 P.AFK:  .ASCII  <19>\    kind = DO (!AD)\
                                  29 44 41 21 28 00490
43 52 55 4F 53 20 3D 20 64 6E 69 6B 20 2C 0F 00495 P.AFL:  .ASCII  <15>\, kind = SOURCE\
                                                  45 004A4
55 4F 53 20 3D 20 64 6E 69 6B 20 20 20 20 17 004A5 P.AFM:  .ASCII  <23>\    kind = SOURCE (!AD)\
                        29 44 41 21 28 20 45 43 52 004B4
53 49 47 45 52 20 3D 20 64 6E 69 6B 20 2C 11 004BD P.AFN:  .ASCII  <17>\, kind = REGISTER\
                                       52 45 54 004CC
43 45 58 45 5C 4E 45 45 52 43 53 47 42 44 1D 004CF P.AFO:  .ASCII  <29>\DBGSCREEN\<92>\EXECUTE_SHODISP_CM\
   4D 43 5F 50 53 49 44 4F 48 53 5F 45 54 55 004DE
                                                  44 004EC        .ASCII  \D\


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                  01FC 00000    .ENTRY  DBG$SCR_EXECUTE_SHODISP_CMD, Save R2,R3,R4,-; 2184
                                                                        R5,R6,R7,R8
                                  58 00000000G   00  9E 00002  MOVAB   DBG$NEWLINE, R8
                                  57 00000000'   EF  9E 00009  MOVAB   DBG$SCR_DISPLAY_LIST, R7
                                  56 00000000G   00  9E 00010  MOVAB   DBG$PRINT, R6
```

```
                55 00000000'   EF  9E 00017          MOVAB   P.AFB, R5                                ;
                50             67  9E 0001E          MOVAB   DBG$SCR_DISPLAY_LIST, R0                ; 2213
                50             67  D1 00021          CMPL    DBG$SCR_DISPLAY_LIST, R0                ;
                               0B  12 00024          BNEQ    1$                                      ;
                               7E  D4 00026          CLRL    -(SP)                                   ; 2216
                               5E  DD 00028          PUSHL   R5                                      ;
                66             02  FB 0002A          CALLS   #2, DBG$PRINT                           ;
                68             00  FB 0002D          CALLS   #0, DBG$NEWLINE                         ; 2217
                               04  00030          RET                                             ; 2215
                52             67  D0 00031 1$:     MOVL    DBG$SCR_DISPLAY_LIST, DISPTR            ; 2225
                50             67  9E 00034 2$:     MOVAB   DBG$SCR_DISPLAY_LIST, R0                ; 2226
                50             52  D1 00037          CMPL    DISPTR, R0                              ;
                               01  12 0003A          BNEQ    3$                                      ;
                               04  0003C          RET                                             ;
                         4C    A2  9F 0003D 3$:     PUSHAB  76(DISPTR)                              ; 2232
                         14    A5  9F 00040         PUSHAB  P.AFC                                   ;
                66             02  FB 00043         CALLS   #2, DBG$PRINT                           ;
                53       08    A7  D0 00046         MOVL    DBG$SCR_WINDOW_LIST, WPTR               ; 2240
                54       10    A2  9E 0004A         MOVAB   16(DISPTR), R4                          ; 2243
                50       08    A7  9E 0004E 4$:     MOVAB   DBG$SCR_WINDOW_LIST, R0                 ; 2241
                50             53  D1 00052         CMPL    WPTR, R0                                ;
                               2D  13 00055         BEQL    6$                                      ;
                64       08    A3  B1 00057         CMPW    8(WPTR), (R4)                           ; 2243
                               22  12 0005B         BNEQ    5$                                      ;
          02    A4       0A    A3  B1 0005D         CMPW    10(WPTR), 2(R4)                         ; 2244
                               1B  12 00062         BNEQ    5$                                      ;
          14    A2       0C    A3  B1 00064         CMPW    12(WPTR), 20(DISPTR)                    ; 2245
                               14  12 00069         BNEQ    5$                                      ;
          16    A2       0E    A3  B1 0006B         CMPW    14(WPTR), 22(DISPTR)                    ; 2246
                               0D  12 00070         BNEQ    5$                                      ;
                         10    A3  9F 00072         PUSHAB  16(WPTR)                                ; 2249
                         20    A5  9F 00075         PUSHAB  P.AFD                                   ;
                66             02  FB 00078         CALLS   #2, DBG$PRINT                           ;
                               53  D4 0007B         CLRL    WPTR                                    ; 2250
                               05  11 0007D         BRB     6$                                      ; 2248
                53             63  D0 0007F 5$:     MOVL    (WPTR), WPTR                            ; 2254
                               CA  11 00082         BRB     4$                                      ; 2241
                               53  D5 00084 6$:     TSTL    WPTR                                    ; 2263
                               2C  13 00086         BEQL    8$                                      ;
                    00000000G  00  95 00088         TSTB    DBG$GL_DEVELOPER                        ; 2266
                               17  18 0008E         BGEQ    7$                                      ;
                7E       16    A2  3C 00090         MOVZWL  22(DISPTR), -(SP)                       ; 2269
                7E       14    A2  3C 00094         MOVZWL  20(DISPTR), -(SP)                       ;
                7E       02    A4  3C 00098         MOVZWL  2(R4), -(SP)                            ; 2268
                7E             64  3C 0009C         MOVZWL  (R4), -(SP)                             ;
                         28    A5  9F 0009F         PUSHAB  P.AFE                                   ; 2267
                66             05  FB 000A2         CALLS   #5, DBG$PRINT                           ;
                               0D  11 000A5         BRB     8$                                      ; 2266
                7E       02    A4  3C 000A7 7$:     MOVZWL  2(R4), -(SP)                            ; 2272
                7E             64  3C 000AB         MOVZWL  (R4), -(SP)                             ;
                         3E    A5  9F 000AE         PUSHAB  P.AFF                                   ; 2271
                66             03  FB 000B1         CALLS   #3, DBG$PRINT                           ;
                7E       1C    A2  3C 000B4 8$:     MOVZWL  28(DISPTR), -(SP)                       ; 2280
                         4C    A5  9F 000B8         PUSHAB  P.AFG                                   ; 2279
                66             02  FB 000BB         CALLS   #2, DBG$PRINT                           ;
                08       0A    A2  E9 000BE         BLBC    10(DISPTR), 9$                          ; 2285
                               7E  D4 000C2         CLRL    -(SP)                                   ; 2287
```

```
                                                59   A5  9F 000C4          PUSHAB  P.AFH                                    ;
                                              66     02  FB 000C7          CALLS   #2, DBG$PRINT                           ;
                                    08      0A    A2  04  E1 000CA 9$:      BBC     #4, 10(DISPTR), 10$                     ; 2289
                                                      7E  D4 000CF          CLRL    -(SP)                                   ; 2291
                                                63    A5  9F 000D1          PUSHAB  P.AFI                                   ;
                                              66      02  FB 000D4          CALLS   #2, DBG$PRINT                           ;
                                    04        00      08  A2  8F 000D7 10$:  CASEB   8(DISPTR), #0, #4                      ; 2297
                          003B      0026        001F      000A      000DC 11$:  .WORD   12$-11$,-                               ;
                                                          0060      000E4               13$-11$,-                           ;
                                                                                        14$-11$,-                           ;
                                                                                        15$-11$,-                           ;
                                                                                        18$-11$                             ;
                                                  00D0    C5  9F 000E6 12$:     PUSHAB  P.AFO                                   ; 2356
                                                      01  DD 000EA           PUSHL   #1                                      ;
                                                00028362  8F  DD 000EC           PUSHL   #164706                                 ;
                                    00000000G   00        03  FB 000F2           CALLS   #3, LIB$SIGNAL                          ;
                                                          4A  11 000F9           BRB     20$                                     ;
                                                      7E  D4 000FB 13$:     CLRL    -(SP)                                   ; 2306
                                                72    A5  9F 000FD          PUSHAB  P.AFJ                                   ;
                                                      40  11 00100          BRB     19$                                     ;
                                              54      48  A2  D0 00102 14$:  MOVL    72(DISPTR), CMDPTR                      ; 2316
                                              68          00  FB 00106          CALLS   #0, DBG$NEWLINE                         ; 2317
                                                    02    A4  9F 00109          PUSHAB  2(CMDPTR)                               ; 2319
                                            7E              64  3C 0010C          MOVZWL  (CMDPTR), -(SP)                         ;
                                                            6E  D7 0010F          DECL    (SP)                                    ;
                                                  0082      C5  9F 00111          PUSHAB  P.AFK                                   ; 2318
                                                            20  11 00115          BRB     17$                                     ; 2319
                                                48    A2  D5 00117 15$:     TSTL    72(DISPTR)                              ; 2329
                                                      08  12 0011A          BNEQ    16$                                     ;
                                                      7E  D4 0011C          CLRL    -(SP)                                   ; 2331
                                                  0096    C5  9F 0011E          PUSHAB  P.AFL                                   ;
                                                          1E  11 00122          BRB     19$                                     ;
                                            54    48    A2  D0 00124 16$:     MOVL    72(DISPTR), CMDPTR                      ; 2337
                                            68          00  FB 00128          CALLS   #0, DBG$NEWLINE                         ; 2338
                                                  02    A4  9F 0012B          PUSHAB  2(CMDPTR)                               ; 2340
                                          7E              64  3C 0012E          MOVZWL  (CMDPTR), -(SP)                         ;
                                                          6E  D7 00131          DECL    (SP)                                    ;
                                                  00A6    C5  9F 00133          PUSHAB  P.AFM                                   ; 2339
                                            66            03  FB 00137 17$:     CALLS   #3, DBG$PRINT                           ; 2340
                                                          09  11 0013A          BRB     20$                                     ; 2297
                                                      7E  D4 0013C 18$:     CLRL    -(SP)                                   ; 2350
                                                  00BE    C5  9F 0013E          PUSHAB  P.AFN                                   ;
                                            66            02  FB 00142 19$:     CALLS   #2, DBG$PRINT                           ;
                                            68            00  FB 00145 20$:     CALLS   #0, DBG$NEWLINE                         ; 2363
                                            52            62  D0 00148          MOVL    (DISPTR), DISPTR                        ; 2364
                                                    FEE6      31  00014B          BRW     2$                                      ; 2226
                                                              04  0014E          RET                                             ; 2373

; Routine Size:  335 bytes,    Routine Base:  DBG$CODE + 1444
```

```
;  2261     2374  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_SHOSEL_CMD(VERB_NODE): NOVALUE =
;  2262     2375  1 !
;  2263     2376  1 ! FUNCTION
;  2264     2377  1 !       This routine executes the SHOW SELECT command.  It accepts the address
;  2265     2378  1 !       of the Verb Node for the command as input and then executes the command.
;  2266     2379  1 !       As there are no command parameters, the Verb Node is actually ignored.
;  2267     2380  1 !       The command simply prints out the display selections for the current
;  2268     2381  1 !       output display, the current scrolling display, and so on.
;  2269     2382  1 !
;  2270     2383  1 ! INPUTS
;  2271     2384  1 !       VERB_NODE - A pointer to the Verb Node for the SHOW SELECT command to
;  2272     2385  1 !                   be executed.  The Verb Node is actually ignored.
;  2273     2386  1 !
;  2274     2387  1 ! OUTPUTS
;  2275     2388  1 !       NONE
;  2276     2389  1 !
;  2277     2390  1
;  2278     2391  2     BEGIN
;  2279     2392  2
;  2280     2393  2     BIND
;  2281     2394  2         NONE_STRING = UPLIT BYTE(%ASCIC 'none');
;  2282     2395  2
;  2283     2396  2     LOCAL
;  2284     2397  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to selected Display Entry
;  2285     2398  2         NAMEPTR: REF VECTOR[,BYTE];     ! Pointer to display name string
;  2286     2399  2
;  2287     2400  2
;  2288     2401  2
;  2289     2402  2     ! If screen mode is not set, say so before giving SELECT settings.
;  2290     2403  2     !
;  2291     2404  2     IF NOT .DBG$GL_SCREEN_MODE
;  2292     2405  2     THEN
;  2293     2406  3         BEGIN
;  2294     2407  3         DBG$PRINT(UPLIT BYTE(%ASCIC 'screen mode not set'), 0);
;  2295     2408  3         DBG$NEWLINE();
;  2296     2409  2         END;
;  2297     2410  2
;  2298     2411  2
;  2299     2412  2     ! Print the header text.
;  2300     2413  2     !
;  2301     2414  2     DBG$PRINT(UPLIT BYTE(%ASCIC 'display selections:'), 0);
;  2302     2415  2     DBG$NEWLINE();
;  2303     2416  2
;  2304     2417  2
;  2305     2418  2     ! Print out the history SELECT setting.
;  2306     2419  2     !
;  2307     2420  2     IF .DBG$GL_SCREEN_HISTORY NEQ 0
;  2308     2421  2     THEN
;  2309     2422  3         BEGIN
;  2310     2423  3         DISPTR = .DBG$GL_SCREEN_HISTORY;
;  2311     2424  3         NAMEPTR = DISPTR[DBG$A_DISP_NAME];
;  2312     2425  3         DBG$PRINT(UPLIT BYTE(%ASCIC '    history= !AC'), .NAMEPTR);
;  2313     2426  3         DBG$NEWLINE();
;  2314     2427  2         END;
;  2315     2428  2
;  2316     2429  2
;  2317     2430  2     ! Print out the input SELECT setting.
```

```
;  2318          2431  2      !
;  2319          2432  2      IF .DBG$SCR_CURDISP_INPUT NEQ 0
;  2320          2433  2      THEN
;  2321          2434  3          BEGIN
;  2322          2435  3          DISPTR = .DBG$SCR_CURDISP_INPUT;
;  2323          2436  3          NAMEPTR = DISPTR[DBG$A_DISP_NAME];
;  2324          2437  3          DBG$PRINT(UPLIT BYTE(%ASCIC '    input  = !AC'), .NAMEPTR);
;  2325          2438  3          DBG$NEWLINE();
;  2326          2439  2          END;
;  2327          2440  2
;  2328          2441  2
;  2329          2442  2      ! Print out the output SELECT setting.
;  2330          2443  2      !
;  2331          2444  2      DISPTR = .DBG$SCR_CURDISP_OUTPUT;
;  2332          2445  2      NAMEPTR = NONE_STRING;
;  2333          2446  2      IF .DISPTR NEQ 0 THEN NAMEPTR = DISPTR[DBG$A_DISP_NAME];
;  2334          2447  2      DBG$PRINT(UPLIT BYTE(%ASCIC '    output = !AC'), .NAMEPTR);
;  2335          2448  2      DBG$NEWLINE();
;  2336          2449  2
;  2337          2450  2
;  2338          2451  2      ! Print out the scrolling SELECT setting.
;  2339          2452  2      !
;  2340          2453  2      DISPTR = .DBG$SCR_CURDISP_SCROLL;
;  2341          2454  2      NAMEPTR = NONE_STRING;
;  2342          2455  2      IF .DISPTR NEQ 0 THEN NAMEPTR = DISPTR[DBG$A_DISP_NAME];
;  2343          2456  2      DBG$PRINT(UPLIT BYTE.%ASCIC '    scroll = !AC'), .NAMEPTR);
;  2344          2457  2      DBG$NEWLINE();
;  2345          2458  2
;  2346          2459  2
;  2347          2460  2      ! Print out the source SELECT setting.
;  2348          2461  2      !
;  2349          2462  2      DISPTR = .DBG$SCR_CURDISP_SOURCE;
;  2350          2463  2      NAMEPTR = NONE_STRING;
;  2351          2464  2      IF .DISPTR NEQ 0 THEN NAMEPTR = DISPTR[DBG$A_DISP_NAME];
;  2352          2465  2      DBG$PRINT(UPLIT BYTE(%ASCIC '    source = !AC'), .NAMEPTR);
;  2353          2466  2      DBG$NEWLINE();
;  2354          2467  2
;  2355          2468  2
;  2356          2469  2      ! All done--now return.
;  2357          2470  2      !
;  2358          2471  2      RETURN;
;  2359          2472  2
;  2360          2473  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                                  65 6E 6F 6E 04  004ED P.AFP:  .ASCII  <4>\none\
6F 6E 20 65 64 6F 6D 20 6E 65                     65 72 63 73 13  004F2 P.AFQ:  .ASCII  <19>\screen mode not set\
                                                  74 65 73 20 74  00501
74 63 65 6C 65 73 20 79 61 6C                     70 73 69 64 13  00506 P.AFR:  .ASCII  <19>\display selections:\
                                                  3A 73 6E 6F 69  00515
21 20 3D 79 72 6F 74 73 69 68                     20 20 20 20 10  0051A P.AFS:  .ASCII  <16>\    history= !AC\
                                                                  43 41  00529
21 20 3D 20 20 74 75 70 6E 69                     20 20 20 20 10  0052B P.AFT:  .ASCII  <16>\    input  = !AC\
                                                                  43 41  0053A
```

```
21 20 3D 20 74 75 70 74 75 6F 20 20 20 20 10 0053C P.AFU:  .ASCII  <16>\    output = !AC\
                                           43 41 0054B
21 20 3D 20 6C 6C 6F 72 63 73 20 20 20 20 10 0054D P.AFV:  .ASCII  <16>\    scroll = !AC\
                                           43 41 0055C
21 20 3D 20 65 63 72 75 6F 73 20 20 20 20 10 0055E P.AFW:  .ASCII  <16>\    source = !AC\
                                           43 41 0056D

                                                  NONE_STRING=        P.AFP

                                                              .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        00FC 00000            .ENTRY  DBG$SCR_EXECUTE_SHOSEL_CMD, Save R2,R3,R4,-  ; 2374
                                                              R5,R6,R7
57 00000000'  EF  9E 00002            MOVAB   DBG$SCR_CURDISP_INPUT, R7
56 00000000G  00  9E 00009            MOVAB   DBG$NEWLINE, R6
55 00000000G  00  9E 00010            MOVAB   DBG$PRINT, R5
54 00000000'  EF  9E 00017            MOVAB   NONE_STRING, R4
0B 00000000'  EF  E8 0001E            BLBS    DBG$GL_SCREEN_MODE, 1$                                    ; 2404
              7E  D4 00025            CLRL    -(SP)                                                     ; 2407
        05    A4  9F 00027            PUSHAB  P.AFQ
65            02  FB 0002A            CALLS   #2, DBG$PRINT
66            00  FB 0002D            CALLS   #0, DBG$NEWLINE                                           ; 2408
              7E  D4 00030 1$:        CLRL    -(SP)                                                     ; 2414
        19    A4  9F 00032            PUSHAB  P.AFR
65            02  FB 00035            CALLS   #2, DBG$PRINT
66            00  FB 00038            CALLS   #0, DBG$NEWLINE                                           ; 2415
50 00000000'  EF  D0 0003B            MOVL    DBG$GL_SCREEN_HISTORY, R0                                 ; 2420
              12  13 00042            BEQL    2$
53            50  D0 00044            MOVL    R0, DISPTR                                                ; 2423
52        4C  A3  9E 00047            MOVAB   76(R3), NAMEPTR                                           ; 2424
              52  DD 0004B            PUSHL   NAMEPTR                                                   ; 2425
        2D    A4  9F 0004D            PUSHAB  P.AFS
65            02  FB 00050            CALLS   #2, DBG$PRINT
66            00  FB 00053            CALLS   #0, DBG$NEWLINE                                           ; 2426
50            67  D0 00056 2$:        MOVL    DBG$SCR_CURDISP_INPUT, R0                                 ; 2432
              12  13 00059            BEQL    3$
53            50  D0 0005B            MOVL    R0, DISPTR                                                ; 2435
52        4C  A3  9E 0005E            MOVAB   76(R3), NAMEPTR                                           ; 2436
              52  DD 00062            PUSHL   NAMEPTR                                                   ; 2437
        3E    A4  9F 00064            PUSHAB  P.AFT
65            02  FB 00067            CALLS   #2, DBG$PRINT
66            00  FB 0006A            CALLS   #0, DBG$NEWLINE                                           ; 2438
53        04  A7  D0 0006D 3$:        MOVL    DBG$SCR_CURDISP_OUTPUT, DISPTR                            ; 2444
52            64  9E 00071            MOVAB   NONE_STRING, NAMEPTR                                      ; 2445
              53  D5 00074            TSTL    DISPTR                                                    ; 2446
              04  13 00076            BEQL    4$
52        4C  A3  9E 00078            MOVAB   76(R3), NAMEPTR
              52  DD 0007C 4$:        PUSHL   NAMEPTR                                                   ; 2447
        4F    A4  9F 0007E            PUSHAB  P.AFU
65            02  FB 00081            CALLS   #2, DBG$PRINT
66            00  FB 00084            CALLS   #0, DBG$NEWLINE                                           ; 2448
53        08  A7  D0 00087            MOVL    DBG$SCR_CURDISP_SCROLL, DISPTR                            ; 2453
52            64  9E 0008B            MOVAB   NONE_STRING, NAMEPTR                                      ; 2454
              53  D5 0008E            TSTL    DISPTR                                                    ; 2455
              04  13 00090            BEQL    5$
52        4C  A3  9E 00092            MOVAB   76(R3), NAMEPTR
```

```
                        52  DD 00096 5$:    PUSHL   NAMEPTR                                 ; 2456
                    60  A4  9F 00098        PUSHAB  P.AFV                                   ;
                65      02  FB 0009B        CALLS   #2, DBG$PRINT                           ;
                66      00  FB 0009E        CALLS   #0, DBG$NEWLINE                         ; 2457
                53  0C  A7  D0 000A1        MOVL    DBG$SCR_CURDISP_SOURCE, DISPTR          ; 2462
                52      64  9E 000A5        MOVAB   NONE_STRING, NAMEPTR                    ; 2463
                        53  D5 000A8        TSTL    DISPTR                                  ; 2464
                        04  13 000AA        BEQL    6$                                      ;
                52  4C  A3  9E 000AC        MOVAB   76(R3), NAMEPTR                         ;
                        52  DD 000B0 6$:    PUSHL   NAMEPTR                                 ; 2465
                    71  A4  9F 000B2        PUSHAB  P.AFW                                   ;
                65      02  FB 000B5        CALLS   #2, DBG$PRINT                           ;
                66      00  FB 000B8        CALLS   #0, DBG$NEWLINE                         ; 2466
                            04 000BB        RET                                             ; 2473

; Routine Size:  188 bytes,    Routine Base:  DBG$CODE + 1593
```

```
;  2362          2474  1 GLOBAL ROUTINE DBG$SCR_EXECUTE_SHOWIND_CMD(VERB_NODE): NOVALUE =
;  2363          2475  1 !
;  2364          2476  1 ! FUNCTION
;  2365          2477  1 !       This routine executes the SHOW WINDOW command.  It accepts the address
;  2366          2478  1 !       of the Verb Node for the command as input and executes the command.
;  2367          2479  1 !
;  2368          2480  1 ! INPUTS
;  2369          2481  1 !       VERB_NODE - A pointer to the Verb Node for the SHOW WINDOW command to
;  2370          2482  1 !                   be executed.
;  2371          2483  1 !
;  2372          2484  1 ! OUTPUTS
;  2373          2485  1 !       NONE
;  2374          2486  1 !
;  2375          2487  1
;  2376          2488  2     BEGIN
;  2377          2489  2
;  2378          2490  2     LOCAL
;  2379          2491  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to current Window Entry
;  2380          2492  2
;  2381          2493  2
;  2382          2494  2
;  2383          2495  2     ! If the Screen Window List is empty, print a message to that effect and
;  2384          2496  2     ! return immediately.
;  2385          2497  2     !
;  2386          2498  2     IF .DBG$SCR_WINDOW_LIST[0] EQL DBG$SCR_WINDOW_LIST
;  2387          2499  2     THEN
;  2388          2500  3         BEGIN
;  2389          2501  3         DBG$PRINT(UPLIT BYTE(%ASCIC 'no windows defined'), 0);
;  2390          2502  3         DBG$NEWLINE();
;  2391          2503  3         RETURN;
;  2392          2504  2         END;
;  2393          2505  2
;  2394          2506  2
;  2395          2507  2     ! Loop through all Window Entries on the Screen Window List and print out
;  2396          2508  2     ! the name and window parameters of each such screen window.
;  2397          2509  2     !
;  2398          2510  2     WPTR = .DBG$SCR_WINDOW_LIST[0];
;  2399          2511  2     WHILE .WPTR NEQ DBG$SCR_WINDOW_LIST DO
;  2400          2512  3         BEGIN
;  2401          2513  3
;  2402          2514  3
;  2403          2515  3         ! Print out the current window's name and window parameters.
;  2404          2516  3         !                                                                    ; 1
;  2405          2517  3         DBG$PRINT(UPLIT BYTE(%ASCIC 'window !AC'), WPTR[DBG$A_WINDOW_NAME]);
;  2406          2518  4         IF .DBG$GL_DEVELOPER[7] THEN BEGIN      !<<--------- TEMPORARY ---------
;  2407          2519  4         DBG$PRINT(UPLIT BYTE(%ASCIC ' at (!SL,!SL,!SL,!SL)'),
;  2408          2520  4                 .WPTR[DBG$W_WINDOW_RBEG], .WPTR[DBG$W_WINDOW_RLEN],
;  2409          2521  4                 .WPTR[DBG$W_WINDOW_CBEG], .WPTR[DBG$W_WINDOW_CLEN]);
;  2410          2522  4         END ELSE BEGIN                          !<<--------- TEMPORARY ---------
;  2411          2523  4         DBG$PRINT(UPLIT BYTE(%ASCIC ' at (!SL,!SL)'),                    !<<-----
;  2412          2524  4                 .WPTR[DBG$W_WINDOW_RBEG], .WPTR[DBG$W_WINDOW_RLEN]);    !<<-----
;  2413          2525  3         END;                                    !<<--------- TEMPORARY ---------
;  2414          2526  3
;  2415          2527  3
;  2416          2528  3         ! Flush out this print line and loop for the next Window Entry.
;  2417          2529  3         !
;  2418          2530  3         DBG$NEWLINE();
```

```
;  2419          2531  3          WPTR = .WPTR[DBG$L_WINDOW_FLINK];
;  2420          2532  3
;  2421          2533  2          END;                                    End of loop over Window List
;  2422          2534  2
;  2423          2535  2
;  2424          2536  2      ! All windows have been shown.  Now return.
;  2425          2537  2      !
;  2426          2538  2      RETURN;
;  2427          2539  2
;  2428          2540  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

66 65 64 20 73 77 6F 64 6E 69 77 20 6F 6E 12  0056F P.AFX:   .ASCII  <18>\no windows defined\                    ;
                                              64 65 6E 69 0057E                                              ;
            43 41 21 20 77 6F 64 6E 69 77 0A  00582 P.AFY:   .ASCII  <10>\window !AC\                         ;
21 2C 4C 53 21 2C 4C 53 21 28 20 74 61 20 15  0058D P.AFZ:   .ASCII  <21>\ at (!SL,!SL,!SL,!SL)\              ;
               29 4C 53 21 2C 4C 53           0059C                                                          ;
   29 4C 53 21 2C 4C 53 21 28 20 74 61 20 0D  005A3 P.AGA:   .ASCII  <13>\ at (!SL,!SL)\                      ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        007C 00000              .ENTRY  DBG$SCR_EXECUTE_SHOWIND_CMD, Save R2,R3,R4,-; 2474
                                                                        R5,R6
                  56 00000000G   00  9E 00002                   MOVAB   DBG$NEWLINE, R6                        ;
                  55 00000000'   EF  9E 00009                   MOVAB   DBG$SCR_WINDOW_LIST, R5                ;
                  54 00000000G   00  9E 00010                   MOVAB   DBG$PRINT, R4                          ;
                  53 00000000'   EF  9E 00017                   MOVAB   P.AFX, R3                              ;
                  50             65  9E 0001E                   MOVAB   DBG$SCR_WINDOW_LIST, R0                ; 2498
                  50             65  D1 00021                   CMPL    DBG$SCR_WINDOW_LIST, R0                ;
                                 0B  12 00024                   BNEQ    1$                                     ;
                                 7E  D4 00026                   CLRL    -(SP)                                  ; 2501
                                 53  DD 00028                   PUSHL   R3                                     ;
                  64             02  FB 0002A                   CALLS   #2, DBG$PRINT                          ;
                  66             00  FB 0002D                   CALLS   #0, DBG$NEWLINE                        ; 2502
                                     04 00030                   RET                                            ; 2500
                  52             65  D0 00031 1$:               MOVL    DBG$SCR_WINDOW_LIST, WPTR              ; 2510
                  50             65  9E 00034 2$:               MOVAB   DBG$SCR_WINDOW_LIST, R0                ; 2511
                  50             52  D1 00037                   CMPL    WPTR, R0                               ;
                                 3F  13 0003A                   BEQL    5$                                     ;
                        10       A2  9F 0003C                   PUSHAB  16(WPTR)                               ; 2517
                        13       A3  9F 0003F                   PUSHAB  P.AFY                                  ;
                  64             02  FB 00042                   CALLS   #2, DBG$PRINT                          ;
                     00000000G   00  95 00045                   TSTB    DBG$GL_DEVELOPER                       ; 2518
                                 18  18 0004B                   BGEQ    3$                                     ;
                  7E       0E    A2  3C 0004D                   MOVZWL  14(WPTR), -(SP)                        ; 2521
                  7E       0C    A2  3C 00051                   MOVZWL  12(WPTR), -(SP)                        ;
                  7E       0A    A2  3C 00055                   MOVZWL  10(WPTR), -(SP)                        ; 2520
                  7E       08    A2  3C 00059                   MOVZWL  8(WPTR), -(SP)                         ;
                           1E    A3  9F 0005D                   PUSHAB  P.AFZ                                  ; 2519
                  64             05  FB 00060                   CALLS   #5, DBG$PRINT                          ;
                                 0E  11 00063                   BRB     4$                                     ; 2518
                  7E       0A    A2  3C 00065 3$:               MOVZWL  10(WPTR), -(SP)                        ; 2524
```

```
                7E      08    A2  3C 00069          MOVZWL  8(WPTR), -(SP)                  ;
                        34    A3  9F 0006D          PUSHAB  P.AGA                           ; 2523
                64            03  FB 00070          CALLS   #3, DBG$PRINT                   ;
                66            00  FB 00073 4$:      CALLS   #0, DBG$NEWLINE                 ; 2530
                52            62  D0 00076          MOVL    (WPTR), WPTR                    ; 2531
                              B9  11 00079          BRB     2$                              ; 2511
                                  04 0007B 5$:      RET                                     ; 2540

; Routine Size:  124 bytes,    Routine Base:  DBG$CODE + 164F
```

```
;2430    2541  1 GLOBAL ROUTINE DBG$SCR_GENERATE_SCREEN(DISPID): NOVALUE =
;2431    2542  1 !
;2432    2543  1 ! FUNCTION
;2433    2544  1 !       This routine is called when all automatically updated screen displays
;2434    2545  1 !       are to have their contents regenerated.  It can also be called to
;2435    2546  1 !       automatically regenerate the contents of a single specified display.
;2436    2547  1 !       The way in which a display's contents is generated depends on the
;2437    2548  1 !       display kind, but for a DO or SOURCE display it it accomplished by
;2438    2549  1 !       adding the display's DEBUG command list to the Command Input Stream
;2439    2550  1 !       in such a way that the output from that command list is redirected
;2440    2551  1 !       to the desired display.
;2441    2552  1 !
;2442    2553  1 ! INPUTS
;2443    2554  1 !       DISPID  - If a single display is to have its contents automatically
;2444    2555  1 !                 regenerated, DISPID is the Display ID (Screen Display Entry
;2445    2556  1 !                 pointer) of that display.  If all automatically updated
;2446    2557  1 !                 displays are to be regenerated, DISPTR is zero.
;2447    2558  1 !
;2448    2559  1 ! OUTPUTS
;2449    2560  1 !       NONE
;2450    2561  1 !
;2451    2562  1
;2452    2563  2     BEGIN
;2453    2564  2
;2454    2565  2     LOCAL
;2455    2566  2         BPTR: REF VECTOR[,WORD],        ! Pointer to new command list buffer
;2456    2567  2         FLINK: REF DBG$DLINE_ENTRY,     ! Pointer to next Error Line Entry
;2457    2568  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to current Screen Display Entry
;2458    2569  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Error Display Line Entry
;2459    2570  2         KIND,                           ! The kind of the current display
;2460    2571  2         TPTR: REF VECTOR[,WORD];        ! Pointer to DEBUG command list buffer
;2461    2572  2
;2462    2573  2
;2463    2574  2
;2464    2575  2     ! Set up a pointer to the first Screen Display Entry whose contents are to
;2465    2576  2     ! be regenerated.  Note that we go through the displays in reverse order
;2466    2577  2     ! since their DEBUG command lists will be stacked on the Command Input
;2467    2578  2     ! Stream and will be executed in Last-In First-Out (LIFO) order.
;2468    2579  2     !
;2469    2580  2     DISPTR = .DISPID;
;2470    2581  2     IF .DISPTR EQL 0 THEN DISPTR = .DBG$SCR_DISPLAY_LIST[1];
;2471    2582  2
;2472    2583  2
;2473    2584  2     ! Now loop for all displays on the Screen Display List to be updated.
;2474    2585  2     !
;2475    2586  2     WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;2476    2587  3         BEGIN
;2477    2588  3
;2478    2589  3
;2479    2590  3         ! If the current display has an Error Line Entry List, throw away all
;2480    2591  3         ! error Display Line Entries on that list.  (These errors apply to the
;2481    2592  3         ! old display contents, not to the new contents we will generate.)
;2482    2593  3         !
;2483    2594  3         DLEPTR = .DISPTR[DBG$L_DISP_ERROR_PTR];
;2484    2595  3         DISPTR[DBG$L_DISP_ERROR_PTR] = 0;
;2485    2596  3         WHILE .DLEPTR NEQ 0 DO
;2486    2597  4             BEGIN
```

```
;  2487          2598  4              FLINK = .DLEPTR[DBG$L_DLINE_FLINK];
;  2488          2599  4              DBG$REL_MEMORY(.DLEPTR);
;  2489          2600  4              DLEPTR = .FLINK;
;  2490          2601  4              END;
;  2491          2602
;  2492          2603
;  2493          2604                 ! If the display still happens to have an old-text list of Display Line
;  2494          2605                 ! Entries, clear out that list too and release all its entries.
;  2495          2606                 !
;  2496          2607                 DLEPTR = .DISPTR[DBG$L_DISP_OLDTXT_PTR];
;  2497          2608                 DISPTR[DBG$L_DISP_OLDTXT_PTR] = 0;
;  2498          2609                 WHILE .DLEPTR NEQ 0 DO
;  2499          2610  4                  BEGIN
;  2500          2611  4                  FLINK = .DLEPTR[DBG$L_DLINE_FLINK];
;  2501          2612  4                  DBG$REL_MEMORY(.DLEPTR);
;  2502          2613  4                  DLEPTR = .FLINK;
;  2503          2614  4                  END;
;  2504          2615
;  2505          2616
;  2506          2617                 ! Case on the kind of the current display.  Update the contents of only
;  2507          2618                 ! those displays that can be automatically updated.  Note that we pre-
;  2508          2619                 ! vent the automatic updating of removed displays by changing KIND.
;  2509          2620                 !
;  2510          2621                 KIND = .DISPTR[DBG$B_DISP_KIND];
;  2511          2622                 IF .DISPTR[DBG$V_DISP_REMOVE] THEN KIND = DBG$K_DISP_NOKIND;
;  2512          2623                 CASE .KIND FROM DBG$K_DISP_MINKIND TO DBG$K_DISP_MAXKIND OF
;  2513          2624                     SET
;  2514          2625
;  2515          2626
;  2516          2627                 ! Handle displays with DO command lists.  Here we add the display's
;  2517          2628                 ! DO command list to the Command Input Stream with output redirected
;  2518          2629                 ! to this display.  Note that we save the current output and source
;  2519          2630                 ! displays in the CIS entry so that print and source output can be
;  2520          2631                 ! directed back to those displays by DBG$NCIS_REMOVE at the end of
;  2521          2632                 ! the command list processing.
;  2522          2633                 !
;  2523          2634                 [DBG$K_DISP_DO]:
;  2524          2635  4                  BEGIN
;  2525          2636  4
;  2526          2637  4
;  2527          2638  4                  ! If changed lines are marked for this display, move the old
;  2528          2639  4                  ! text of the display to the old-text list.  The old-text list
;  2529          2640  4                  ! is made into a singly-linked, zero-terminated list.  The
;  2530          2641  4                  ! new display contents are then cleared to zero lines.
;  2531          2642  4                  !
;  2532          2643  4                  IF .DISPTR[DBG$V_DISP_MARKFLG] AND
;  2533          2644  5                     (.DISPTR[DBG$W_DISP_LINECNT] NEQ 0)
;  2534          2645  4                  THEN
;  2535          2646  5                      BEGIN
;  2536          2647  5                      DISPTR[DBG$L_DISP_OLDTXT_PTR] =
;  2537          2648  5                                          .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  2538          2649  5                      DLEPTR = .DISPTR[DBG$L_DISP_END_LINE_PTR];
;  2539          2650  5                      DLEPTR[DBG$L_DLINE_FLINK] = 0;
;  2540          2651  5                      DISPTR[DBG$L_DISP_START_LINE_PTR] =
;  2541          2652  5                                          DISPTR[DBG$L_DISP_START_LINE_PTR];
;  2542          2653  5                      DISPTR[DBG$L_DISP_END_LINE_PTR] =
;  2543          2654  5                                          DISPTR[DBG$L_DISP_START_LINE_PTR];
```

```
;  2544          2655  5                 DISPTR[DBG$W_DISP_LINECNT] = 0;
;  2545          2656  5                 DISPTR[DBG$L_DISP_WINDOW_PTR] = 0;
;  2546          2657  5                 DISPTR[DBG$W_DISP_DROW] = 1;
;  2547          2658  5                 DISPTR[DBG$W_DISP_SCROLL] = 0;
;  2548          2659  5                 DISPTR[DBG$V_DISP_INVSCR] = TRUE;
;  2549          2660  5                 END
;  2550          2661  5
;  2551          2662  5
;  2552          2663  5             ! Otherwise, simply empty the display of its present contents.
;  2553          2664  5             !
;  2554          2665  4             ELSE
;  2555          2666  4                 DBG$SCR_EMPTY_DISPLAY(.DISPTR);
;  2556          2667  4
;  2557          2668  4
;  2558          2669  4             ! Create a copy of the DEBUG command list, add it to the Command
;  2559          2670  4             ! Input Stream, and redirect output to the present display.
;  2560          2671  4             !
;  2561          2672  4             TPTR = .DISPTR[DBG$L_DISP_CMDLIST];
;  2562          2673  4             BPTR = DBG$GET_MEMORY((.TPTR[0]+5)/%UPVAL);
;  2563          2674  4             CH$MOVE(.TPTR[0] + 2, TPTR[0], BPTR[0]);
;  2564          2675  4             DBG$NCIS_ADD(BPTR[1], .BPTR[0], DBG$K_CIS_SCREEN, 0, 0, 0);
;  2565          2676  4             DBG$GL_CISHEAD[CIS$L_SCREEN_OUTPUT] = .DBG$GL_SCREEN_OUTPUT;
;  2566          2677  4             DBG$GL_CISHEAD[CIS$L_SCREEN_SOURCE] = .DBG$GL_SCREEN_SOURCE;
;  2567          2678  4             DBG$GL_CISHEAD[CIS$L_SCREEN_ERROR] = .DBG$GL_SCREEN_ERROR;
;  2568          2679  4             DBG$GL_CISHEAD[CIS$V_SCREEN_NOGO] = .DBG$GL_SCREEN_NOGO;
;  2569          2680  4             DBG$GL_SCREEN_OUTPUT = .DISPTR;
;  2570          2681  4             DBG$GL_SCREEN_SOURCE = 0;
;  2571          2682  4             DBG$GL_SCREEN_ERROR = .DISPTR;
;  2572          2683  4             DBG$GL_SCREEN_NOGO = TRUE;
;  2573          2684  3             END;
;  2574          2685  3
;  2575          2686  3
;  2576          2687  3             ! Handle displays with SOURCE command lists.  (If the SOURCE display
;  2577          2688  3             ! has no command list, we do nothing.)  Here we add the display's
;  2578          2689  3             ! DEBUG command list to the Command Input Stream with source output
;  2579          2690  3             ! redirected to this display.  Note that we save the current output
;  2580          2691  3             ! and source displays in the CIS entry so that print and source
;  2581          2692  3             ! output can be directed back to those displays by DBG$NCIS_REMOVE
;  2582          2693  3             ! at the end of the command list processing.
;  2583          2694  3             !
;  2584          2695  3             [DBG$K_DISP_SOURCE]:
;  2585          2696  4                 BEGIN
;  2586          2697  4                 TPTR = .DISPTR[DBG$L_DISP_CMDLIST];
;  2587          2698  4                 IF .TPTR NEQ 0
;  2588          2699  4                 THEN
;  2589          2700  5                     BEGIN
;  2590          2701  5                     BPTR = DBG$GET_MEMORY((.TPTR[0]+5)/%UPVAL);
;  2591          2702  5                     CH$MOVE(.TPTR[0] + 2, TPTR[0], BPTR[0]);
;  2592          2703  5                     DBG$NCIS_ADD(BPTR[1], .BPTR[0], DBG$K_CIS_SCREEN, 0, 0, 0);
;  2593          2704  5                     DBG$GL_CISHEAD[CIS$L_SCREEN_OUTPUT] = .DBG$GL_SCREEN_OUTPUT;
;  2594          2705  5                     DBG$GL_CISHEAD[CIS$L_SCREEN_SOURCE] = .DBG$GL_SCREEN_SOURCE;
;  2595          2706  5                     DBG$GL_CISHEAD[CIS$L_SCREEN_ERROR] = .DBG$GL_SCREEN_ERROR;
;  2596          2707  5                     DBG$GL_CISHEAD[CIS$V_SCREEN_NOGO] = .DBG$GL_SCREEN_NOGO;
;  2597          2708  5                     DBG$GL_SCREEN_OUTPUT = .DBG$SCR_CURDISP_OUTPUT;
;  2598          2709  5                     DBG$GL_SCREEN_SOURCE = .DISPTR;
;  2599          2710  5                     DBG$GL_SCREEN_ERROR = .DISPTR;
;  2600          2711  5                     DBG$GL_SCREEN_NOGO = TRUE;
```

```
;  2601          2712  4                    END;
;  2602          2713  4
;  2603          2714  3                END;
;  2604          2715  3
;  2605          2716  3
;  2606          2717  3                ! Handle REGISTER displays.  Here we regenerate the display text
;  2607          2718  3                ! using the current register values.
;  2608          2719  3                !
;  2609          2720  3                [DBG$K_DISP_REGISTER]:
;  2610          2721  3                    REGISTER_DISPLAY(.DISPTR, DBG$RUNFRAME[DBG$L_USER_REGS]);
;  2611          2722  3
;  2612          2723  3
;  2613          2724  3                ! Any other display kind cannot be regenerated, so we do nothing.
;  2614          2725  3                ! Note that we get here for any removed displays as well.
;  2615          2726  3                !
;  2616          2727  3                [INRANGE, OUTRANGE]:
;  2617          2728  3                    0;
;  2618          2729  3
;  2619          2730  3                TES;
;  2620          2731  3
;  2621          2732  3
;  2622          2733  3            ! Loop for the next display unless only a single display was to be
;  2623          2734  3            ! regenerated.  Note that we loop in reverse order as mentioned above.
;  2624          2735  3            !
;  2625          2736  3            IF .DISPID NEQ 0 THEN EXITLOOP;
;  2626          2737  3            DISPTR = .DISPTR[DBG$L_DISP_BLINK];
;  2627          2738  3
;  2628          2739  2            END;                            ! End of loop over displays
;  2629          2740  2
;  2630          2741  2
;  2631          2742  2        ! All requested displays have been regenerated.  Now return.
;  2632          2743  2        !
;  2633          2744  2        RETURN;
;  2634          2745  2
;  2635          2746  1        END;


                                     0FFC 00000          .ENTRY  DBG$SCR_GENERATE_SCREEN, Save R2,R3,R4,R5,- ; 2541
                                                                 R6,R7,R8,R9,R10,R11
                     56        04    AC   D0 00002          MOVL    DISPID, DISPTR                              ; 2580
                                     07   12 00006          BNEQ    1$                                          ; 2581
                     56 00000000'    EF   D0 00008          MOVL    DBG$SCR_DISPLAY_LIST+4, DISPTR
                     50 00000000'    EF   9E 0000F 1$:      MOVAB   DBG$SCR_DISPLAY_LIST, R0                    ; 2586
                     50              56   D1 00016          CMPL    DISPTR, R0
                                     01   12 00019          BNEQ    2$
                                          04 0001B          RET
                     57        2C    A6   D0 0001C 2$:      MOVL    44(DISPTR), DLEPTR                          ; 2594
                               2C    A6   D4 00020          CLRL    44(DISPTR)                                  ; 2595
                                     57   D5 00023 3$:      TSTL    DLEPTR                                      ; 2596
                                     11   13 00025          BEQL    4$
                     5A              67   D0 00027          MOVL    (DLEPTR), FLINK                             ; 2598
                                     57   DD 0002A          PUSHL   DLEPTR                                      ; 2599
00000000G            00              01   FB 0002C          CALLS   #1, DBG$REL_MEMORY
                     57              5A   D0 00033          MOVL    FLINK, DLEPTR                               ; 2600
```

```
                                                EB  11 00036          BRB     3$                                                      ; 2596
                                        57      30  A6 DO 00038 4$:   MOVL    48(DISPTR), DLEPTR                                      ; 2607
                                        30      A6  D4 0003C          CLRL    48(DISPTR)                                              ; 2608
                                                57  D5 0003F 5$:      TSTL    DLEPTR                                                  ; 2609
                                                11  13 00041          BEQL    6$                                                      ;
                                        5A              67  D0 00043 MOVL    (DLEPTR), FLINK                                         ; 2611
                                                57  DD 00046          PUSHL   DLEPTR                                                  ; 2612
                                00000000G       00  01 FB 00048      CALLS   #1, DBG$REL_MEMORY                                      ;
                                        57      5A  D0 0004F          MOVL    FLINK, DLEPTR                                           ; 2613
                                                EB  11 00052          BRB     5$                                                      ; 2609
                                        5B      08  A6 9A 00054 6$:   MOVZBL  8(DISPTR), KIND                                         ; 2621
                                        02      0A  A6 E9 00058       BLBC    10(DISPTR), 7$                                          ; 2622
                                                5B  D4 0005C          CLRL    KIND                                                    ;
                                04              00  5B CF 0005E 7$:   CASEL   KIND, #0, #4                                           ; 2623
                        00AA    000D            0130    0130  00062 8$:   .WORD   16$-8$,-                                           ;
                                                        0123  0006A           16$-8$,-                                                ;
                                                                              9$-8$,-                                                 ;
                                                                              12$-8$,-                                                ;
                                                                              15$-8$                                                  ;
                                                    0123  31 0006C          BRW     16$                                             ;
                                        2D      0A  A6      04  E1 0006F 9$:   BBC     #4, 10(DISPTR), 10$                         ; 2643
                                                    1E  A6  B5 00074          TSTW    30(DISPTR)                                      ; 2644
                                                        28  13 00077          BEQL    10$                                             ;
                                                30  A6      20  A6 D0 00079  MOVL    32(DISPTR), 48(DISPTR)                          ; 2648
                                                    24  A6  D0 0007E          MOVL    36(DISPTR), DLEPTR                              ; 2649
                                                        67  D4 00082          CLRL    (DLEPTR)                                        ; 2650
                                                20  A6      20  A6 9E 00084  MOVAB   32(DISPTR), 32(DISPTR)                         ; 2652
                                                24  A6      20  A6 9E 00089  MOVAB   32(DISPTR), 36(DISPTR)                         ; 2654
                                                    1E  A6  B4 0008E          CLRW    30(DISPTR)                                      ; 2655
                                                    28  A6  D4 00091          CLRL    40(DISPTR)                                      ; 2656
                                                18  A6              01  B0 00094 MOVW    #1, 24(DISPTR)                          ; 2657
                                                    0C  A6  B4 00098          CLRW    12(DISPTR)                                      ; 2658
                                                0A  A6              02  88 0009B  BISB2   #2, 10(DISPTR)                          ; 2659
                                                                    07  11 0009F  BRB     11$                                     ; 2643
                                                                    56  DD 000A1 10$:   PUSHL   DISPTR                             ; 2666
                                            F6F0    CF              01  FB 000A3  CALLS   #1, DBG$SCR_EMPTY_DISPLAY                 ;
                                                    59      48  A6  D0 000A8 11$:   MOVL    72(DISPTR), TPTR                        ; 2672
                                                    50          69  3C 000AC      MOVZWL  (TPTR), R0                              ; 2673
                                                    50          05  C0 000AF      ADDL2   #5, R0                                  ;
                                        7E          50          04  C7 000B2      DIVL3   #4, R0, -(SP)                           ;
                                00000000G           00          01  FB 000B6      CALLS   #1, DBG$GET_MEMORY                      ;
                                                    58          50  D0 000BD      MOVL    R0, BPTR                                ;
                                                    50          69  3C 000C0      MOVZWL  (TPTR), R0                              ; 2674
                                                    50          02  C0 000C3      ADDL2   #2, R0                                  ;
                                        68          69          50  28 000C6      MOVC3   R0, (TPTR), (BPTR)                      ;
                                                                7E  7C 000CA      CLRQ    -(SP)                                   ; 2675
                                                    7E          08  7D 000CC      MOVQ    #8, -(SP)                               ;
                                                    7E          68  3C 000CF      MOVZWL  (BPTR), -(SP)                           ;
                                                            02      A8  9F 000D2  PUSHAB  2(BPTR)                                 ;
                                00000000G           00          06  FB 000D5      CALLS   #6, DBG$NCIS_ADD                        ;
                                                    50  00000000G  00  D0 000DC      MOVL    DBG$GL_CISHEAD, R0                      ; 2676
                                                24  A0  00000000' EF  7D 000E3      MOVQ    DBG$GL_SCREEN_OUTPUT, 36(R0)            ;
                                                2C  A0  00000000' EF  D0 000EB      MOVL    DBG$GL_SCREEN_ERROR, 44(R0)             ; 2678
                        12      A0      01          02  00000000' EF  F0 000F3  INSV    DBG$GL_SCREEN_NOGO, #2, #1, 18(R0)      ; 2679
                                        00000000'   EF              56  D0 000FD  MOVL    DISPTR, DBG$GL_SCREEN_OUTPUT            ; 2680
                                                            00000000' EF  D4 00104  CLRL    DBG$GL_SCREEN_SOURCE                    ; 2681
                                                                        69  11 0010A  BRB     13$                                     ; 2682
```

```
                                59        48   A6  D0 0010C 12$:    MOVL    72(DISPTR), TPTR                        ; 2697
                                                71  13 00110        BEQL    14$                                     ; 2698
                                50              69  3C 00112        MOVZWL  (TPTR), R0                              ; 2701
                                50              05  C0 00115        ADDL2   #5, R0
                             7E                 50  C7 00118        DIVL3   #4, R0, -(SP)
                          00000000G             00  FB 0011C        CALLS   #1, DBG$GET_MEMORY
                                58              50  D0 00123        MOVL    R0, BPTR
                                50              69  3C 00126        MOVZWL  (TPTR), R0                              ; 2702
                                50              02  C0 00129        ADDL2   #2, R0
                             68                 69  50 28 0012C     MOVC3   R0, (TPTR), (BPTR)
                                                7E  7C 00130        CLRQ    -(SP)                                   ; 2703
                                7E              08  7D 00132        MOVQ    #8, -(SP)
                                7E              68  3C 00135        MOVZWL  (BPTR), -(SP)
                                         02     A8  9F 00138        PUSHAB  2(BPTR)
                          00000000G             00  FB 0013B        CALLS   #6, DBG$NCIS_ADD
                                50 00000000G    00  D0 00142        MOVL    DBG$GL_CISHEAD, R0                      ; 2704
                             24 A0 00000000'    EF  7D 00149        MOVQ    DBG$GL_SCREEN_OUTPUT, 36(R0)
                             2C A0 00000000'    EF  D0 00151        MOVL    DBG$GL_SCREEN_ERROR, 44(R0)             ; 2706
    12   A0                   01 02 00000000'    EF  F0 00159        INSV    DBG$GL_SCREEN_NOGO, #2, #1, 18(R0)      ; 2707
                          00000000' EF 00000000'    EF  D0 00163        MOVL    DBG$SCR_CURDISP_OUTPUT, -               ; 2708
                                                                                DBG$GL_SCREEN_OUTPUT
                          00000000' EF              56  D0 0016E        MOVL    DISPTR, DBG$GL_SCREEN_SOURCE            ; 2709
                          00000000' EF              56  D0 00175 13$:   MOVL    DISPTR, DBG$GL_SCREEN_ERROR             ; 2710
                          00000000' EF              01  D0 0017C        MOVL    #1, DBG$GL_SCREEN_NOGO                  ; 2711
                                                    0D  11 00183 14$:   BRB     16$                                     ; 2623
                                00000000G           00  9F 00185 15$:   PUSHAB  DBG$RUNFRAME+4                         ; 2721
                                                    56  DD 00188        PUSHL   DISPTR
                                   0000V            CF  02 FB 0018D     CALLS   #2, REGISTER_DISPLAY
                                                 04     AC  D5 00192 16$:   TSTL    DISPID                          ; 2736
                                                    07  12 00195        BNEQ    17$
                                56              04     A6  D0 00197        MOVL    4(DISPTR), DISPTR                   ; 2737
                                                   FE71 31 0019B        BRW     1$                                  ; 2586
                                                    04  00 0019E 17$:   RET                                         ; 2746

; Routine Size:  415 bytes,    Routine Base:  DBG$CODE + 16CB
```

```
;  2637          2747  1 GLOBAL ROUTINE DBG$SCR_INITIALIZE: NOVALUE =
;  2638          2748  1 !
;  2639          2749  1 ! FUNCTION
;  2640          2750  1 !       This routine initializes the Screen Debugging code.  It must be called
;  2641          2751  1 !       before any other Screen Debugging routines are called in order to set
;  2642          2752  1 !       up or initialize various data structures and locations needed by those
;  2643          2753  1 !       other routines.  This routine should be called exactly once.
;  2644          2754  1 !
;  2645          2755  1 ! INPUTS
;  2646          2756  1 !       NONE
;  2647          2757  1 !
;  2648          2758  1 ! OUTPUTS
;  2649          2759  1 !       NONE
;  2650          2760  1 !
;  2651          2761  1
;  2652          2762  2     BEGIN
;  2653          2763  2
;  2654          2764  2
;  2655          2765  2
;  2656          2766  2     ! Initialize the Screen Display List and the Screen Window List list heads
;  2657          2767  2     ! so that they constitute empty doubly linked lists.  This is done at run-
;  2658          2768  2     ! time to keep DEBUG's code position-independent (PIC).
;  2659          2769  2     !
;  2660          2770  2     DBG$SCR_DISPLAY_LIST[0] = DBG$SCR_DISPLAY_LIST;
;  2661          2771  2     DBG$SCR_DISPLAY_LIST[1] = DBG$SCR_DISPLAY_LIST;
;  2662          2772  2     DBG$SCR_WINDOW_LIST[0] = DBG$SCR_WINDOW_LIST;
;  2663          2773  2     DBG$SCR_WINDOW_LIST[1] = DBG$SCR_WINDOW_LIST;
;  2664          2774  2
;  2665          2775  2
;  2666          2776  2     ! Create the predefined screen window definitions.
;  2667          2777  2     !
;  2668          2778  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'FS'),   1, 19, 1, 132);
;  2669          2779  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'H1'),   1,  9, 1, 132);
;  2670          2780  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'H2'),  11,  9, 1, 132);
;  2671          2781  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'T1'),   1,  6, 1, 132);
;  2672          2782  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'T2'),   8,  6, 1, 132);
;  2673          2783  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'T3'),  15,  5, 1, 132);
;  2674          2784  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'T12'),  1, 13, 1, 132);
;  2675          2785  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'T23'),  8, 12, 1, 132);
;  2676          2786  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q1'),   1,  4, 1, 132);
;  2677          2787  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q2'),   6,  4, 1, 132);
;  2678          2788  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q3'),  11,  4, 1, 132);
;  2679          2789  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q4'),  16,  4, 1, 132);
;  2680          2790  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q123'), 1, 14, 1, 132);
;  2681          2791  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q234'), 6, 14, 1, 132);
;  2682          2792  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q23'),  6,  9, 1, 132);
;  2683          2793  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'R1'),   1,  5, 1, 132);
;  2684          2794  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'R2'),   7,  5, 1, 132);
;  2685          2795  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'R3'),  13,  5, 1, 132);
;  2686          2796  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'R12'),  1, 11, 1, 132);
;  2687          2797  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'R23'),  7, 13, 1, 132);
;  2688          2798  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'H12'),  1, 19, 1, 132);
;  2689          2799  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'T123'), 1, 19, 1, 132);
;  2690          2800  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q1234'),1, 19, 1, 132);
;  2691          2801  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q12'),  1,  9, 1, 132);
;  2692          2802  2     DBG$SCR_CREATE_WINDOW(UPLIT BYTE(%ASCIC 'Q34'), 11,  9, 1, 132);
;  2693          2803  2
```

```
;  2694          2804  2
;  2695          2805  2      ! Mark all lines in the "old" screen image as invalid.  This ensures that
;  2696          2806  2      ! DEBUG won't make any assumptions about the previous screen contents when
;  2697          2807  2      ! first putting displays on the screen.
;  2698          2808  2      !
;  2699          2809  2      INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  2700          2810  2          OLD_VALID[.I] = FALSE;
;  2701          2811  2
;  2702          2812  2
;  2703          2813  2      ! The initialization is complete.   Return to the caller.
;  2704          2814  2      !
;  2705          2815  2      RETURN;
;  2706          2816  2
;  2707          2817  1      END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                    53  46  02  005B1 P.AGB:  .ASCII  <2>\FS\                                    ;
                                    31  48  02  005B4 P.AGC:  .ASCII  <2>\H1\                                    ;
                                    32  48  02  005B7 P.AGD:  .ASCII  <2>\H2\                                    ;
                                    31  54  02  005BA P.AGE:  .ASCII  <2>\T1\                                    ;
                                    32  54  02  005BD P.AGF:  .ASCII  <2>\T2\                                    ;
                                    33  54  02  005C0 P.AGG:  .ASCII  <2>\T3\                                    ;
                                32  31  54  03  005C3 P.AGH:  .ASCII  <3>\T12\                                   ;
                                33  32  54  03  005C7 P.AGI:  .ASCII  <3>\T23\                                   ;
                                    31  51  02  005CB P.AGJ:  .ASCII  <2>\Q1\                                    ;
                                    32  51  02  005CE P.AGK:  .ASCII  <2>\Q2\                                    ;
                                    33  51  02  005D1 P.AGL:  .ASCII  <2>\Q3\                                    ;
                                    34  51  02  005D4 P.AGM:  .ASCII  <2>\Q4\                                    ;
                                33  32  31  51  04  005D7 P.AGN:  .ASCII  <4>\Q123\                               ;
                                34  33  32  51  04  005DC P.AGO:  .ASCII  <4>\Q234\                               ;
                                    33  32  51  03  005E1 P.AGP:  .ASCII  <3>\Q23\                               ;
                                        31  52  02  005E5 P.AGQ:  .ASCII  <2>\R1\                                ;
                                        32  52  02  005E8 P.AGR:  .ASCII  <2>\R2\                                ;
                                        33  52  02  005EB P.AGS:  .ASCII  <2>\R3\                                ;
                                    32  31  52  03  005EE P.AGT:  .ASCII  <3>\R12\                               ;
                                    33  32  52  03  005F2 P.AGU:  .ASCII  <3>\R23\                               ;
                                    32  31  48  03  005F6 P.AGV:  .ASCII  <3>\H12\                               ;
                                33  32  31  54  04  005FA P.AGW:  .ASCII  <4>\T123\                              ;
                            34  33  32  31  51  05  005FF P.AGX:  .ASCII  <5>\Q1234\                              ;
                                    32  31  51  03  00605 P.AGY:  .ASCII  <3>\Q12\                               ;
                                    34  33  51  03  00609 P.AGZ:  .ASCII  <3>\Q34\                               ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                            001C 00000          .ENTRY  DBG$SCR_INITIALIZE, Save R2,R3,R4           ; 2747
                                54 00000000'  EF  9E 00002      MOVAB   DBG$SCR_DISPLAY_LIST, R4                    ;
                                53     F2DF   CF  9E 00009      MOVAB   DBG$SCR_CREATE_WINDOW, R3                   ;
                                52 00000000'  EF  9E 0000E      MOVAB   P.AGB, R2                                   ;
                                64            64  9E 00015      MOVAB   DBG$SCR_DISPLAY_LIST, DBG$SCR_DISPLAY_LIST  ; 2770
                            04  A4            64  9E 00018      MOVAB   DBG$SCR_DISPLAY_LIST, -                     ; 2771
                                                                    DBG$SCR_DISPLAY_LIST+4
                            08  A4        08  A4  9E 0001C      MOVAB   DBG$SCR_WINDOW_LIST, DBG$SCR_WINDOW_LIST    ; 2772
```

```
                0C  A4      08  A4  9E 00021         MOVAB   DBG$SCR_WINDOW_LIST, DBG$SCR_WINDOW_LIST+4 ;2773
                    7E      84  8F  9A 00026         MOVZBL  #132, -(SP)                                 ;2778
                                01  DD 0002A         PUSHL   #1
                                13  DD 0002C         PUSHL   #19
                                01  DD 0002E         PUSHL   #1
                                52  DD 00030         PUSHL   R2
                            63      05  FB 00032     CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 00035         MOVZBL  #132, -(SP)                                 ;2779
                                01  DD 00039         PUSHL   #1
                                09  DD 0003B         PUSHL   #9
                                01  DD 0003D         PUSHL   #1
                            03  A2  9F 0003F         PUSHAB  P.AGC
                            63      05  FB 00042     CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 00045         MOVZBL  #132, -(SP)                                 ;2780
                                01  DD 00049         PUSHL   #1
                                09  DD 0004B         PUSHL   #9
                                0B  DD 0004D         PUSHL   #11
                            06  A2  9F 0004F         PUSHAB  P.AGD
                            63      05  FB 00052     CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 00055         MOVZBL  #132, -(SP)                                 ;2781
                                01  DD 00059         PUSHL   #1
                                06  DD 0005B         PUSHL   #6
                                01  DD 0005D         PUSHL   #1
                            09  A2  9F 0005F         PUSHAB  P.AGE
                            63      05  FB 00062     CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 00065         MOVZBL  #132, -(SP)                                 ;2782
                                01  DD 00069         PUSHL   #1
                                06  DD 0006B         PUSHL   #6
                                08  DD 0006D         PUSHL   #8
                            0C  A2  9F 0006F         PUSHAB  P.AGF
                        63      05  FB 00072         CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 00075         MOVZBL  #132, -(SP)                                 ;2783
                                01  DD 00079         PUSHL   #1
                                05  DD 0007B         PUSHL   #5
                                0F  DD 0007D         PUSHL   #15
                            0F  A2  9F 0007F         PUSHAB  P.AGG
                        63      05  FB 00082         CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 00085         MOVZBL  #132, -(SP)                                 ;2784
                                01  DD 00089         PUSHL   #1
                                0D  DD 0008B         PUSHL   #13
                                01  DD 0008D         PUSHL   #1
                            12  A2  9F 0008F         PUSHAB  P.AGH
                        63      05  FB 00092         CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 00095         MOVZBL  #132, -(SP)                                 ;2785
                                01  DD 00099         PUSHL   #1
                                0C  DD 0009B         PUSHL   #12
                                08  DD 0009D         PUSHL   #8
                            16  A2  9F 0009F         PUSHAB  P.AGI
                        63      05  FB 000A2         CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 000A5         MOVZBL  #132, -(SP)                                 ;2786
                                01  DD 000A9         PUSHL   #1
                                04  DD 000AB         PUSHL   #4
                                01  DD 000AD         PUSHL   #1
                            1A  A2  9F 000AF         PUSHAB  P.AGJ
                        63      05  FB 000B2         CALLS   #5, DBG$SCR_CREATE_WINDOW
                    7E      84  8F  9A 000B5         MOVZBL  #132, -(SP)                                 ;2787
                                01  DD 000B9         PUSHL   #1
```

```
                04  DD 000BB          PUSHL   #4
                06  DD 000BD          PUSHL   #6
             1D A2  9F 000BF          PUSHAB  P.AGK
          63    05  FB 000C2          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 000C5          MOVZBL  #132, -(SP)                     ;2788
                01  DD 000C9          PUSHL   #1
                04  DD 000CB          PUSHL   #4
                0B  DD 000CD          PUSHL   #11
             20 A2  9F 000CF          PUSHAB  P.AGL
          63    05  FB 000D2          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 000D5          MOVZBL  #132, -(SP)                     ;2789
                01  DD 000D9          PUSHL   #1
                04  DD 000DB          PUSHL   #4
                10  DD 000DD          PUSHL   #16
             23 A2  9F 000DF          PUSHAB  P.AGM
          63    05  FB 000E2          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 000E5          MOVZBL  #132, -(SP)                     ;2790
                01  DD 000E9          PUSHL   #1
                0E  DD 000EB          PUSHL   #14
                01  DD 000ED          PUSHL   #1
             26 A2  9F 000EF          PUSHAB  P.AGN
          63    05  FB 000F2          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 000F5          MOVZBL  #132, -(SP)                     ;2791
                01  DD 000F9          PUSHL   #1
                0E  DD 000FB          PUSHL   #14
                06  DD 000FD          PUSHL   #6
             2B A2  9F 000FF          PUSHAB  P.AGO
          63    05  FB 00102          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 00105          MOVZBL  #132, -(SP)                     ;2792
                01  DD 00109          PUSHL   #1
                09  DD 0010B          PUSHL   #9
                06  DD 0010D          PUSHL   #6
             30 A2  9F 0010F          PUSHAB  P.AGP
          63    05  FB 00112          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 00115          MOVZBL  #132, -(SP)                     ;2793
                01  DD 00119          PUSHL   #1
                05  DD 0011B          PUSHL   #5
                01  DD 0011D          PUSHL   #1
             34 A2  9F 0011F          PUSHAB  P.AGQ
          63    05  FB 00122          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 00125          MOVZBL  #132, -(SP)                     ;2794
                01  DD 00129          PUSHL   #1
                05  DD 0012B          PUSHL   #5
                07  DD 0012D          PUSHL   #7
             37 A2  9F 0012F          PUSHAB  P.AGR
          63    05  FB 00132          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 00135          MOVZBL  #132, -(SP)                     ;2795
                01  DD 00139          PUSHL   #1
                05  DD 0013B          PUSHL   #5
                0D  DD 0013D          PUSHL   #13
             3A A2  9F 0013F          PUSHAB  P.AGS
          63    05  FB 00142          CALLS   #5, DBG$SCR_CREATE_WINDOW
          7E 84 8F  9A 00145          MOVZBL  #132, -(SP)                     ;2796
                01  DD 00149          PUSHL   #1
                0B  DD 0014B          PUSHL   #11
                01  DD 0014D          PUSHL   #1
             3D A2  9F 0014F          PUSHAB  P.AGT
```

```
                63              05  FB 00152          CALLS   #5, DBG$SCR_CREATE_WINDOW
7E      84      8F              9A 00155           MOVZBL  #132, -(SP)                     ; 2797
                01              DD 00159           PUSHL   #1
                0D              DD 0015B           PUSHL   #13
                07              DD 0015D           PUSHL   #7
        41      A2              9F 0015F           PUSHAB  P.AGU
                63              05  FB 00162          CALLS   #5, DBG$SCR_CREATE_WINDOW
7E      84      8F              9A 00165           MOVZBL  #132, -(SP)                     ; 2798
                01              DD 00169           PUSHL   #1
                13              DD 0016B           PUSHL   #19
                01              DD 0016D           PUSHL   #1
        45      A2              9F 0016F           PUSHAB  P.AGV
                63              05  FB 00172          CALLS   #5, DBG$SCR_CREATE_WINDOW
7E      84      8F              9A 00175           MOVZBL  #132, -(SP)                     ; 2799
                01              DD 00179           PUSHL   #1
                13              DD 0017B           PUSHL   #19
                01              DD 0017D           PUSHL   #1
        49      A2              9F 0017F           PUSHAB  P.AGW
                63              05  FB 00182          CALLS   #5, DBG$SCR_CREATE_WINDOW
7E      84      8F              9A 00185           MOVZBL  #132, -(SP)                     ; 2800
                01              DD 00189           PUSHL   #1
                13              DD 0018B           PUSHL   #19
                01              DD 0018D           PUSHL   #1
        4E      A2              9F 0018F           PUSHAB  P.AGX
                63              05  FB 00192          CALLS   #5, DBG$SCR_CREATE_WINDOW
7E      84      8F              9A 00195           MOVZBL  #132, -(SP)                     ; 2801
                01              DD 00199           PUSHL   #1
                09              DD 0019B           PUSHL   #9
                01              DD 0019D           PUSHL   #1
        54      A2              9F 0019F           PUSHAB  P.AGY
                63              05  FB 001A2          CALLS   #5, DBG$SCR_CREATE_WINDOW
7E      84      8F              9A 001A5           MOVZBL  #132, -(SP)                     ; 2802
                01              DD 001A9           PUSHL   #1
                09              DD 001AB           PUSHL   #9
                0B              DD 001AD           PUSHL   #11
        58      A2              9F 001AF           PUSHAB  P.AGZ
                63              05  FB 001B2          CALLS   #5, DBG$SCR_CREATE_WINDOW
                50              D4 001B5           CLRL    I                               ; 2809
00      1634    C4              50  E5 001B7  1$:    BBCC    I, OLD_VALID, 2$            ; 2810
F6              50              14  F3 001BD  2$:    AOBLEQ  #20, I, 1$
                                04 001C1           RET                                     ; 2817

; Routine Size:  450 bytes,    Routine Base:  DBG$CODE + 186A
```

```
;  2709          2818  1 ROUTINE DBG$SCR_LOOKUP_DISPLAY(NAMEPTR) =
;  2710          2819  1 !
;  2711          2820  1 ! FUNCTION
;  2712          2821  1 !       This routine looks up the Screen Display Entry which corresponds to a
;  2713          2822  1 !       specified display name.  It accepts a Counted ASCII display name as
;  2714          2823  1 !       input and returns a pointer to the corresponding Screen Display Entry
;  2715          2824  1 !       as output.  If no display by that name exists, zero is returned.
;  2716          2825  1 !
;  2717          2826  1 !       This routine accepts several "pseudo-display" names, all of which
;  2718          2827  1 !       begin with a percent sign.  The pseudo-displays accepted are:
;  2719          2828  1 !
;  2720          2829  1 !               %CURDISP        - The last display on the Screen Display List.
;  2721          2830  1 !                                 (The least occluded display.)
;  2722          2831  1 !               %NEXTDISP       - The first display on the Screen Display List.
;  2723          2832  1 !                                 (The most occluded display.)
;  2724          2833  1 !               %NEXTSCROLL     - The next display on the Screen Display List
;  2725          2834  1 !                                 after the current scrolling display.
;  2726          2835  1 !               %NEXTOUTPUT     - The next output display on the Screen Display
;  2727          2836  1 !                                 List after the current output display.
;  2728          2837  1 !               %NEXTSOURCE     - The next source display on the Screen Display
;  2729          2838  1 !                                 List after the current source display.
;  2730          2839  1 !
;  2731          2840  1 !       If the input display name is a pseudo-display name, it gets converted
;  2732          2841  1 !       into a pointer to the Screen Display Entry of the actual display that
;  2733          2842  1 !       matches the pseudo-display specification.  This is done by searching
;  2734          2843  1 !       the Screen Display List for the best match, always ignoring removed
;  2735          2844  1 !       displays in the search.
;  2736          2845  1 !
;  2737          2846  1 ! INPUTS
;  2738          2847  1 !       NAMEPTR - A pointer to the name of the display to be looked up.  The
;  2739          2848  1 !                 name is represented as a Counted ASCII string.
;  2740          2849  1 !
;  2741          2850  1 ! OUTPUTS
;  2742          2851  1 !       A pointer to the Screen Display Entry of the named display is returned
;  2743          2852  1 !       as the routine value.  (This pointer constitutes the display's
;  2744          2853  1 !       Display ID.)  If no display by the specified name exists, zero
;  2745          2854  1 !       is returned.
;  2746          2855  1 !
;  2747          2856  1
;  2748          2857  2     BEGIN
;  2749          2858  2
;  2750          2859  2     MAP
;  2751          2860  2         NAMEPTR: REF VECTOR[,BYTE];     ! Pointer to ASCIC display name
;  2752          2861  2
;  2753          2862  2     LOCAL
;  2754          2863  2         DISPNAME: REF VECTOR[,BYTE],    ! Pointer to current display's name
;  2755          2864  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to current Display Entry
;  2756          2865  2         FIRST_DISP,                     ! Pointer to first display matching
;  2757          2866  2                                         !      pseudo-display on display list
;  2758          2867  2         PAST_CURRENT;                   ! Flag set if we are past current
;  2759          2868  2                                         !      scrolling, etc. display in
;  2760          2869  2                                         !      the display list
;  2761          2870  2
;  2762          2871  2
;  2763          2872  2
;  2764          2873  2     ! See if the display name begins with a percent sign "%".  If so, this is a
;  2765          2874  2     ! "pseudo-display", meaning that we must search the display list for the
```

```
;  2766     2875  2      ! next display of a certain kind as determined by the pseudo-display name.
;  2767     2876  2      !
;  2768     2877  2      IF .NAMEPTR[0] EQL 0 THEN RETURN 0;
;  2769     2878  2      IF .NAMEPTR[1] EQL '%'
;  2770     2879  2      THEN
;  2771     2880  3          BEGIN
;  2772     2881  3
;  2773     2882  3
;  2774     2883  3          ! If the %CURDISP pseudo-display is requested, we return a pointer to
;  2775     2884  3          ! the last (the most recent) display on the display list.  Note that we
;  2776     2885  3          ! scan the Screen Display List backwards to find the most recent non-
;  2777     2886  3          ! removed display.
;  2778     2887  3          !
;  2779     2888  3          IF CH$EQL(.NAMEPTR[0], NAMEPTR[1], 8, UPLIT BYTE(%ASCII '%CURDISP'), 0)
;  2780     2889  3          THEN
;  2781     2890  4              BEGIN
;  2782     2891  4              DISPTR = .DBG$SCR_DISPLAY_LIST[1];
;  2783     2892  4              WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;  2784     2893  5                  BEGIN
;  2785     2894  5                  IF NOT .DISPTR[DBG$V_DISP_REMOVE] THEN RETURN .DISPTR;
;  2786     2895  5                  DISPTR = .DISPTR[DBG$L_DISP_BLINK];
;  2787     2896  4                  END;
;  2788     2897  4
;  2789     2898  4              RETURN 0;
;  2790     2899  3              END;
;  2791     2900  3
;  2792     2901  3
;  2793     2902  3          ! If the %NEXTDISP pseudo-display is requested, we return a pointer to
;  2794     2903  3          ! the first display on the display list.  This is the "next" display
;  2795     2904  3          ! after the last (most recent) display on the list.
;  2796     2905  3          !
;  2797     2906  3          IF CH$EQL(.NAMEPTR[0], NAMEPTR[1], 9, UPLIT BYTE(%ASCII '%NEXTDISP'), 0)
;  2798     2907  3          THEN
;  2799     2908  4              BEGIN
;  2800     2909  4              DISPTR = .DBG$SCR_DISPLAY_LIST[0];
;  2801     2910  4              WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;  2802     2911  5                  BEGIN
;  2803     2912  5                  IF NOT .DISPTR[DBG$V_DISP_REMOVE] THEN RETURN .DISPTR;
;  2804     2913  5                  DISPTR = .DISPTR[DBG$L_DISP_FLINK];
;  2805     2914  4                  END;
;  2806     2915  4
;  2807     2916  4              RETURN 0;
;  2808     2917  3              END;
;  2809     2918  3
;  2810     2919  3
;  2811     2920  3          ! If this is the %NEXTSCROLL pseudo-display, we find the next display
;  2812     2921  3          ! on the display list after the current scrolling display.  We make a
;  2813     2922  3          ! single scan of the display list, picking either the first display
;  2814     2923  3          ! after the current scrolling display or, if that does not exist, the
;  2815     2924  3          ! first display on the list.  In effect, the search wraps around at the
;  2816     2925  3          ! end of the list.
;  2817     2926  3          !
;  2818     2927  3          IF CH$EQL(.NAMEPTR[0], NAMEPTR[1],
;  2819     2928  3                                  11, UPLIT BYTE(%ASCII '%NEXTSCROLL'), 0)
;  2820     2929  3          THEN
;  2821     2930  4              BEGIN
;  2822     2931  4              FIRST_DISP = 0;
```

```
;2823     2932  4              PAST_CURRENT = FALSE;
;2824     2933  4              DISPTR = .DBG$SCR_DISPLAY_LIST[0];
;2825     2934  4              WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;2826     2935  5                  BEGIN
;2827     2936  5                  IF NOT .DISPTR[DBG$V_DISP_REMOVE]
;2828     2937  5                  THEN
;2829     2938  6                      BEGIN
;2830     2939  6                      IF .FIRST_DISP EQL 0 THEN FIRST_DISP = .DISPTR;
;2831     2940  6                      IF .PAST_CURRENT THEN RETURN .DISPTR;
;2832     2941  5                      END;
;2833     2942  5
;2834     2943  5                  IF .DISPTR EQL .DBG$SCR_CURDISP_SCROLL THEN PAST_CURRENT = TRUE;
;2835     2944  5                  DISPTR = .DISPTR[DBG$L_DISP_FLINK];
;2836     2945  4                  END;
;2837     2946  4
;2838     2947  4              RETURN .FIRST_DISP;
;2839     2948  3              END;
;2840     2949  3
;2841     2950  3
;2842     2951  3          ! If this is the %NEXTOUTPUT pseudo-display, we find the next NORMAL
;2843     2952  3          ! or DO display on the display list after the current output display.
;2844     2953  3          ! We make a single scan of the display list, picking either the first
;2845     2954  3          ! appropriate display after the current output display or the first
;2846     2955  3          ! appropriate display on the list.  In effect, the search wraps around
;2847     2956  3          ! at the end of the list.
;2848     2957  3          !
;2849     2958  3          IF CH$EQL(.NAMEPTR[0], NAMEPTR[1],
;2850     2959  3                                  11, UPLIT BYTE(%ASCII '%NEXTOUTPUT'), 0)
;2851     2960  3          THEN
;2852     2961  4              BEGIN
;2853     2962  4              FIRST_DISP = 0;
;2854     2963  4              PAST_CURRENT = FALSE;
;2855     2964  4              DISPTR = .DBG$SCR_DISPLAY_LIST[0];
;2856     2965  4              WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;2857     2966  5                  BEGIN
;2858     2967  5                  IF (NOT .DISPTR[DBG$V_DISP_REMOVE]) AND
;2859     2968  6                     ((.DISPTR[DBG$B_DISP_KIND] EQL DBG$K_DISP_NORMAL) OR
;2860     2969  6                      (.DISPTR[DBG$B_DISP_KIND] EQL DBG$K_DISP_DO))
;2861     2970  5                  THEN
;2862     2971  6                      BEGIN
;2863     2972  6                      IF .FIRST_DISP EQL 0 THEN FIRST_DISP = .DISPTR;
;2864     2973  6                      IF .PAST_CURRENT THEN RETURN .DISPTR;
;2865     2974  5                      END;
;2866     2975  5
;2867     2976  5                  IF .DISPTR EQL .DBG$SCR_CURDISP_OUTPUT THEN PAST_CURRENT = TRUE;
;2868     2977  5                  DISPTR = .DISPTR[DBG$L_DISP_FLINK];
;2869     2978  4                  END;
;2870     2979  4
;2871     2980  4              RETURN .FIRST_DISP;
;2872     2981  3              END;
;2873     2982  3
;2874     2983  3
;2875     2984  3          ! If this is the %NEXTSOURCE pseudo-display, we find the next SOURCE
;2876     2985  3          ! display on the display list after the current source output display.
;2877     2986  3          ! We make a single scan of the display list, picking either the first
;2878     2987  3          ! appropriate display after the current output display or the first
;2879     2988  3          ! appropriate display on the list.  In effect, the search wraps around
```

```
;2880     2989  3          ! at the end of the list.
;2881     2990  3          !
;2882     2991  3          IF CH$EQL(.NAMEPTR[0], NAMEPTR[1],
;2883     2992  3                                  11, UPLIT BYTE(%ASCII '%NEXTSOURCE'), 0)
;2884     2993  3          THEN
;2885     2994  4              BEGIN
;2886     2995  4              FIRST_DISP = 0;
;2887     2996  4              PAST_CURRENT = FALSE;
;2888     2997  4              DISPTR = .DBG$SCR_DISPLAY_LIST[0];
;2889     2998  4              WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;2890     2999  5                  BEGIN
;2891     3000  5                  IF (NOT .DISPTR[DBG$V_DISP_REMOVE]) AND
;2892     3001  6                     (.DISPTR[DBG$B_DISP_KIND] EQL DBG$K_DISP_SOURCE)
;2893     3002  5                  THEN
;2894     3003  6                      BEGIN
;2895     3004  6                      IF .FIRST_DISP EQL 0 THEN FIRST_DISP = .DISPTR;
;2896     3005  6                      IF .PAST_CURRENT THEN RETURN .DISPTR;
;2897     3006  5                      END;
;2898     3007  5
;2899     3008  5                  IF .DISPTR EQL .DBG$SCR_CURDISP_SOURCE THEN PAST_CURRENT = TRUE;
;2900     3009  5                  DISPTR = .DISPTR[DBG$L_DISP_FLINK];
;2901     3010  4                  END;
;2902     3011  4
;2903     3012  4              RETURN .FIRST_DISP;
;2904     3013  3              END;
;2905     3014  3
;2906     3015  3
;2907     3016  3          ! If we got any other pseudo-display name, we return zero to indicate
;2908     3017  3          ! that no such display exists.
;2909     3018  3          !
;2910     3019  3          RETURN 0;
;2911     3020  3
;2912     3021  2          END;                                ! End of %pseudo-display code
;2913     3022  2
;2914     3023  2
;2915     3024  2          ! The display name is a normal name and does not begin with a percent sign.
;2916     3025  2          ! Loop through all the Display Entries in the Screen Display List until a
;2917     3026  2          ! Display Entry with the desired name is found.  When and if such a display
;2918     3027  2          ! is found, return a pointer to that Display Entry.
;2919     3028  2          !
;2920     3029  2          DISPTR = .DBG$SCR_DISPLAY_LIST[0];
;2921     3030  2          WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;2922     3031  3              BEGIN
;2923     3032  3              DISPNAME = DISPTR[DBG$A_DISP_NAME];
;2924     3033  3              IF CH$EQL(.NAMEPTR[0], NAMEPTR[1], .DISPNAME[0], DISPNAME[1], 0)
;2925     3034  3              THEN
;2926     3035  3                  RETURN .DISPTR;
;2927     3036  3
;2928     3037  3              DISPTR = .DISPTR[DBG$L_DISP_FLINK];
;2929     3038  2              END;
;2930     3039  2
;2931     3040  2
;2932     3041  2          ! There is no display by the specified name.  Return zero.
;2933     3042  2          !
;2934     3043  2          RETURN 0;
;2935     3044  2
;2936     3045  1          END;
```

```
                                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                50  53  49  44  52  55  43  25  0060D P.AHA:  .ASCII  \%CURDISP\                    ;
                            50  53  49  44  54  58  45  4E  25  00615 P.AHB:  .ASCII  \%NEXTDISP\                   ;
                        4C  4C  4F  52  43  53  54  58  45  4E  25  0061E P.AHC:  .ASCII  \%NEXTSCROLL\               ;
                        54  55  50  54  55  4F  54  58  45  4E  25  00629 P.AHD:  .ASCII  \%NEXTOUTPUT\              ;
                        45  43  52  55  4F  53  54  58  45  4E  25  00634 P.AHE:  .ASCII  \%NEXTSOURCE\              ;


                                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                        07FC 00000 DBG$SCR_LOOKUP_DISPLAY:
                                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10        ; 2818
                                    5A 00000000'    EF  9E 00002    MOVAB   P.AHA, R10                              ;
                                    59 00000000'    EF  9E 00009    MOVAB   DBG$SCR_DISPLAY_LIST, R9                ;
                                    57      04      AC  D0 00010    MOVL    NAMEPTR, R7                             ; 2877
                                                    67  95 00014    TSTB    (R7)                                    ;
                                                    42  13 00016    BEQL    5$                                      ;
                                    58      01      A7  9E 00018    MOVAB   1(R7), R8                               ; 2878
                                    25              68  91 0001C    CMPB    (R8), #37                               ;
                                                    03  13 0001F    BEQL    1$                                      ;
                                                00FC    31 00021    BRW     27$                                     ;
                                    50              67  9A 00024 1$:  MOVZBL  (R7), R0                            ; 2888
                        08              00          68  5C  2D 00027    CMPC5   R0, (R8), #0, #8, P.AHA                 ;
                                                        0A      0002C                                               ;
                                                    16  12 0002D    BNEQ    3$                                      ;
                                    54      04      A9  D0 0002F    MOVL    DBG$SCR_DISPLAY_LIST+4, DISPTR          ; 2891
                                    50              69  9E 00033 2$:  MOVAB   DBG$SCR_DISPLAY_LIST, R0              ; 2892
                                    50              54  D1 00036    CMPL    DISPTR, R0                              ;
                                                    1F  13 00039    BEQL    5$                                      ;
                                    20      0A      A4  E9 0003B    BLBC    10(DISPTR), 6$                          ; 2894
                                    54      04      A4  D0 0003F    MOVL    4(DISPTR), DISPTR                       ; 2895
                                                    EE  11 00043    BRB     2$                                      ; 2892
                                    50              67  9A 00045 3$:  MOVZBL  (R7), R0                            ; 2906
                        09              00          68  50  2D 00048    CMPC5   R0, (R8), #0, #9, P.AHB                 ;
                                                        08  AA  0004D                                               ;
                                                    1A  12 0004F    BNEQ    9$                                      ;
                                    54              69  D0 00051    MOVL    DBG$SCR_DISPLAY_LIST, DISPTR            ; 2909
                                    50              69  9E 00054 4$:  MOVAB   DBG$SCR_DISPLAY_LIST, R0              ; 2910
                                    50              54  D1 00057    CMPL    DISPTR, R0                              ;
                                                    03  12 0005A 5$:  BNEQ    6$                                      ;
                                                00E8    31 0005C    BRW     31$                                     ;
                                    03      0A      A4  E8 0005F 6$:  BLBS    10(DISPTR), 8$                          ; 2912
                                                00D8    31 00063 7$:  BRW     29$                                     ;
                                    54              64  D0 00066 8$:  MOVL    (DISPTR), DISPTR                        ; 2913
                                                    E9  11 00069    BRB     4$                                      ; 2910
                                    50              67  9A 0006B 9$:  MOVZBL  (R7), R0                            ; 2927
                        0B              00          68  50  2D 0006E    CMPC5   R0, (R8), #0, #11, P.AHC                ;
                                                        11  AA  00073                                               ;
                                                    29  12 00075    BNEQ    14$                                     ;
                                                    55  7C 00077    CLRQ    FIRST_DISP                              ; 2931
                                    54              69  D0 00079    MOVL    DBG$SCR_DISPLAY_LIST, DISPTR            ; 2933
                                    50              69  9E 0007C 10$:  MOVAB   DBG$SCR_DISPLAY_LIST, R0              ; 2934
```

```
                50              54  D1 0007F          CMPL    DISPTR, R0                                      ;
                                74  13 00082          BEQL    22$                                             ;
                0A      0A      A4  E8 00084          BLBS    10(DISPTR), 12$                                 ; 2936
                                55  D5 00088          TSTL    FIRST_DISP                                      ; 2939
                                03  12 0008A          BNEQ    11$                                             ;
                55              54  D0 0008C          MOVL    DISPTR, FIRST_DISP                              ;
                D1              56  E8 0008F 11$:     BLBS    PAST_CURRENT, 7$                                ; 2940
            F8  A9              54  D1 00092 12$:     CMPL    DISPTR, DBG$SCR_CURDISP_SCROLL                  ; 2943
                                03  12 00096          BNEQ    13$                                             ;
                56              01  D0 00098          MOVL    #1, PAST_CURRENT                                ;
                54              64  D0 0009B 13$:     MOVL    (DISPTR), DISPTR                                ; 2944
                                DC  11 0009E          BRB     10$                                             ; 2934
                50              67  9A 000A0 14$:     MOVZBL  (R7), R0                                        ; 2958
    08      00  68              50  2D 000A3          CMPC5   R0, (R8), #0, #11, P.AHD                        ;
                        1C      AA      000A8                                                                  ;
                                35  12 000AA          BNEQ    20$                                             ;
                                55  7C 000AC          CLRQ    FIRST_DISP                                      ; 2962
                54              69  D0 000AE          MOVL    DBG$SCR_DISPLAY_LIST, DISPTR                    ; 2964
                50              69  9E 000B1 15$:     MOVAB   DBG$SCR_DISPLAY_LIST, R0                        ; 2965
                50              54  D1 000B4          CMPL    DISPTR, R0                                      ;
                                63  13 000B7          BEQL    26$                                             ;
                16      0A      A4  E8 000B9          BLBS    10(DISPTR), 18$                                 ; 2967
                01      08      A4  91 000BD          CMPB    8(DISPTR), #1                                   ; 2968
                                06  13 000C1          BEQL    16$                                             ;
                02      08      A4  91 000C3          CMPB    8(DISPTR), #2                                   ; 2969
                                0A  12 000C7          BNEQ    18$                                             ;
                                55  D5 000C9 16$:     TSTL    FIRST_DISP                                      ; 2972
                                03  12 000CB          BNEQ    17$                                             ;
                55              54  D0 000CD          MOVL    DISPTR, FIRST_DISP                              ;
                6B              56  E8 000D0 17$:     BLBS    PAST_CURRENT, 29$                               ; 2973
            F4  A9              54  D1 000D3 18$:     CMPL    DISPTR, DBG$SCR_CURDISP_OUTPUT                  ; 2976
                                03  12 000D7          BNEQ    19$                                             ;
                56              01  D0 000D9          MOVL    #1, PAST_CURRENT                                ;
                54              64  D0 000DC 19$:     MOVL    (DISPTR), DISPTR                                ; 2977
                                D0  11 000DF          BRB     15$                                             ; 2965
                50              67  9A 000E1 20$:     MOVZBL  (R7), R0                                        ; 2991
    08      00  68              50  2D 000E4          CMPC5   R0, (R8), #0, #11, P.AHE                        ;
                        27      AA      000E9                                                                  ;
                                5A  12 000EB          BNEQ    31$                                             ;
                                55  7C 000ED          CLRQ    FIRST_DISP                                      ; 2995
                54              69  D0 000EF          MOVL    DBG$SCR_DISPLAY_LIST, DISPTR                    ; 2997
                50              69  9E 000F2 21$:     MOVAB   DBG$SCR_DISPLAY_LIST, R0                        ; 2998
                50              54  D1 000F5          CMPL    DISPTR, R0                                      ;
                                22  13 000F8 22$:     BEQL    26$                                             ;
                10      0A      A4  E8 000FA          BLBS    10(DISPTR), 24$                                 ; 3000
                03      08      A4  91 000FE          CMPB    8(DISPTR), #3                                   ; 3001
                                0A  12 00102          BNEQ    24$                                             ;
                                55  D5 00104          TSTL    FIRST_DISP                                      ; 3004
                                03  12 00106          BNEQ    23$                                             ;
                55              54  D0 00108          MOVL    DISPTR, FIRST_DISP                              ;
                30              56  E8 0010B 23$:     BLBS    PAST_CURRENT, 29$                               ; 3005
            FC  A9              54  D1 0010E 24$:     CMPL    DISPTR, DBG$SCR_CURDISP_SOURCE                  ; 3008
                                03  12 00112          BNEQ    25$                                             ;
                56              01  D0 00114          MOVL    #1, PAST_CURRENT                                ;
                54              64  D0 00117 25$:     MOVL    (DISPTR), DISPTR                                ; 3009
                                D6  11 0011A          BRB     21$                                             ; 2998
                50              55  D0 0011C 26$:     MOVL    FIRST_DISP, R0                                  ; 3012
```

```
                                                 04 0011F             RET
                                  54             69 D0 00120 27$:     MOVL    DBG$SCR_DISPLAY_LIST, DISPTR                    ;3029
                                  50             69 9E 00123 28$:     MOVAB   DBG$SCR_DISPLAY_LIST, R0                        ;3030
                                  50             54 D1 00126          CMPL    DISPTR, R0
                                                 1C 13 00129          BEQL    31$
                                  55         4C  A4 9E 0012B          MOVAB   76(R4), DISPNAME                                ;3032
                                  51             67 9A 0012F          MOVZBL  (R7), R1                                        ;3033
                                  50             65 9A 00132          MOVZBL  (DISPNAME), R0
                50                00             68 51 2D 00135       CMPC5   R1, (R8), #0, R0, 1(DISPNAME)
                                                    01 A5    0013A
                                                 04 12 0013C          BNEQ    30$
                                  50             54 D0 0013E 29$:     MOVL    DISPTR, R0                                      ;3035
                                                 04 00141             RET
                                  54             64 D0 00142 30$:     MOVL    (DISPTR), DISPTR                                ;3037
                                                 DC 11 00145          BRB     28$                                             ;3030
                                                 50 D4 00147 31$:     CLRL    R0                                              ;3045
                                                 04 00149             RET

; Routine Size:  330 bytes,    Routine Base:  DBG$CODE + 1A2C
```

```
;  2938     3046  1 ROUTINE DBG$SCR_LOOKUP_WINDOW(NAMEPTR) =
;  2939     3047  1 !
;  2940     3048  1 ! FUNCTION
;  2941     3049  1 !       This routine looks up the Screen Window Entry which corresponds to a
;  2942     3050  1 !       specified window name.  It accepts a Counted ASCII window name as
;  2943     3051  1 !       input and returns a pointer to the corresponding Screen Window Entry
;  2944     3052  1 !       as output.  If no window by that name exists, zero is returned.
;  2945     3053  1 !
;  2946     3054  1 ! INPUTS
;  2947     3055  1 !       NAMEPTR - A pointer to the name of the window to be looked up.  The
;  2948     3056  1 !                 name is represented as a Counted ASCII string.
;  2949     3057  1 !
;  2950     3058  1 ! OUTPUTS
;  2951     3059  1 !       A pointer to the Screen Window Entry of the named window is returned as
;  2952     3060  1 !       the routine value.  If no window by the specified name exists,
;  2953     3061  1 !       zero is returned.
;  2954     3062  1 !
;  2955     3063  1
;  2956     3064  2     BEGIN
;  2957     3065  2
;  2958     3066  2     MAP
;  2959     3067  2         NAMEPTR: REF VECTOR[,BYTE];     ! Pointer to the ASCIC window name
;  2960     3068  2
;  2961     3069  2     LOCAL
;  2962     3070  2         WNAME: REF VECTOR[,BYTE],       ! Pointer to current window's name
;  2963     3071  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to current Window Entry
;  2964     3072  2
;  2965     3073  2
;  2966     3074  2
;  2967     3075  2     ! Loop through all the Window Entries in the Screen Window List until a
;  2968     3076  2     ! Window Entry with the desired name is found.  When and if such a window
;  2969     3077  2     ! is found, return a pointer to that Window Entry.
;  2970     3078  2     !
;  2971     3079  2     WPTR = DBG$SCR_WINDOW_LIST;
;  2972     3080  2     WPTR = .WPTR[DBG$L_WINDOW_FLINK];
;  2973     3081  2     WHILE .WPTR NEQ DBG$SCR_WINDOW_LIST DO
;  2974     3082  3         BEGIN
;  2975     3083  3         WNAME = WPTR[DBG$A_WINDOW_NAME];
;  2976     3084  3         IF CH$EQL(.NAMEPTR[0], NAMEPTR[1], .WNAME[0], WNAME[1], 0)
;  2977     3085  3         THEN
;  2978     3086  3             RETURN .WPTR;
;  2979     3087  3
;  2980     3088  3         WPTR = .WPTR[DBG$L_WINDOW_FLINK];
;  2981     3089  2         END;
;  2982     3090  2
;  2983     3091  2
;  2984     3092  2     ! There is no window by the specified name.  Return zero.
;  2985     3093  2     !
;  2986     3094  2     RETURN 0;
;  2987     3095  2
;  2988     3096  1     END;


                                00FC 00000 DBG$SCR_LOOKUP_WINDOW:
```

```
                                57 00000000'   EF  9E 00002          .WORD   Save R2,R3,R4,R5,R6,R7                  ; 3046
                                54             67  9E 00009          MOVAB   DBG$SCR_WINDOW_LIST, R7
                                54             64  D0 0000C          MOVAB   DBG$SCR_WINDOW_LIST, WPTR               ; 3079
                                54             64  D0 0000C          MOVL    (WPTR), WPTR                            ; 3080
                                56       04    AC  D0 0000F          MOVL    NAMEPTR, R6                             ; 3084
                                50             67  9E 00013 1$:      MOVAB   DBG$SCR_WINDOW_LIST, R0                 ; 3081
                                50             54  D1 00016          CMPL    WPTR, R0
                                               1E  13 00019          BEQL    3$
                                55       10    A4  9E 0001B          MOVAB   16(R4), WNAME                           ; 3083
                                51       04    BC  9A 0001F          MOVZBL  @NAMEPTR, R1                            ; 3084
                                50             65  9A 00023          MOVZBL  (WNAME), R0
        50              00      01    A6             51  2D 00026          CMPC5   R1, 1(R6), #0, R0, 1(WNAME)
                                                01    A5     0002C
                                               04  12 0002E          BNEQ    2$
                                50             54  D0 00030          MOVL    WPTR, R0                                ; 3086
                                               04  00033          RET
                                54             64  D0 00034 2$:      MOVL    (WPTR), WPTR                            ; 3088
                                               DA  11 00037          BRB     1$                                      ; 3081
                                               50  D4 00039 3$:      CLRL    R0                                      ; 3094
                                                   04 0003B          RET                                             ; 3096

; Routine Size:  60 bytes,    Routine Base:  DBG$CODE + 1B76
```

```
;  2990          3097  1 ROUTINE DBG$SCR_OUTPUT_LINE_MINIMAL(NEWLENGTH, NEWTEXT, NEWREND,
;  2991          3098  1                                     OLDLENGTH, OLDTEXT, OLDREND, ROW): NOVALUE =
;  2992          3099  1 !
;  2993          3100  1 ! FUNCTION
;  2994          3101  1 !       This routine outputs a line of text to a specified line on the terminal
;  2995          3102  1 !       screen.  The routine accepts both the new text (including rendition
;  2996          3103  1 !       attributes) and the old text currently on the screen as inputs.  It
;  2997          3104  1 !       then uses a minimal screen update algorithm to update the terminal
;  2998          3105  1 !       text with a minimal (more of less) number of characters being sent to
;  2999          3106  1 !       the terminal.
;  3000          3107  1 !
;  3001          3108  1 ! INPUTS
;  3002          3109  1 !       NEWLENGTH - The length in characters of the new text line to be output
;  3003          3110  1 !                   to the screen.
;  3004          3111  1 !
;  3005          3112  1 !       NEWTEXT - A pointer to the text of the new line to be output to the
;  3006          3113  1 !                 screen.
;  3007          3114  1 !
;  3008          3115  1 !       NEWREND - A pointer to the rendition vector for the new text to be
;  3009          3116  1 !                 output to the screen.  This vector contains one byte of
;  3010          3117  1 !                 rendition bits for each character in the NEWTEXT vector.
;  3011          3118  1 !
;  3012          3119  1 !       OLDLENGTH - The length of the old text line currently on the screen.
;  3013          3120  1 !
;  3014          3121  1 !       OLDTEXT - A pointer to the old text currently on the screen where the
;  3015          3122  1 !                 new text line will be written.  If the old text on the screen
;  3016          3123  1 !                 is unknown, the OLDTEXT pointer must be zero.
;  3017          3124  1 !
;  3018          3125  1 !       OLDREND - A pointer to the rendition vector for the old text currently
;  3019          3126  1 !                 on the screen.
;  3020          3127  1 !
;  3021          3128  1 !       ROW     - The row (line number on the screen) of the line to be updated.
;  3022          3129  1 !
;  3023          3130  1 ! OUTPUTS
;  3024          3131  1 !       NONE
;  3025          3132  1 !
;  3026          3133  1
;  3027          3134  2     BEGIN
;  3028          3135  2
;  3029          3136  2     MAP
;  3030          3137  2         NEWTEXT: REF VECTOR[,BYTE],     ! Pointer to text of new screen line
;  3031          3138  2         NEWREND: REF VECTOR[,BYTE],     ! Pointer to renditions of new line
;  3032          3139  2         OLDTEXT: REF VECTOR[,BYTE],     ! Pointer to old text of line on screen
;  3033          3140  2         OLDREND: REF VECTOR[,BYTE];     ! Pointer to old renditions on screen
;  3034          3141  2
;  3035          3142  2     LOCAL
;  3036          3143  2         OUTEND,                         ! End index of string to output
;  3037          3144  2         OUTSTART,                       ! Start index of string to output
;  3038          3145  2         REND,                           ! Rendition codes for current section
;  3039          3146  2         STATUS,                         ! Status code from screen routines
;  3040          3147  2         TEXT_DESC: BLOCK[8,BYTE],       ! Output string descriptor
;  3041          3148  2         VALIDCOUNT;                     ! Number of valid characters in a row
;  3042          3149  2                                         !      on the screen (in the old text)
;  3043          3150  2
;  3044          3151  2
;  3045          3152  2
;  3046          3153  2     ! Initialize the output line string descriptor.
```

```
;3047     3154  2      !
;3048     3155  2      TEXT_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;3049     3156  2      TEXT_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;3050     3157  2
;3051     3158  2
;3052     3159  2      ! If the old screen contents is completely unknown (indicated by OLDTEXT
;3053     3160  2      ! being zero), the line is erased and then rewritten in its entirety.
;3054     3161  2      ! After that, we return.
;3055     3162  2      !
;3056     3163  2      IF .OLDTEXT EQL 0
;3057     3164  2      THEN
;3058     3165  3          BEGIN
;3059     3166  3
;3060     3167  3
;3061     3168  3          ! Erase the current screen contents of the line.  If the new text has
;3062     3169  3          ! zero length, we then return right away.
;3063     3170  3          !
;3064     3171  3          STATUS = SCR$SET_CURSOR(.ROW, 1);
;3065     3172  3          IF NOT .STATUS THEN SIGNAL(.STATUS);
;3066     3173  3          STATUS = SCR$ERASE_LINE();
;3067     3174  3          IF NOT .STATUS THEN SIGNAL(.STATUS);
;3068     3175  3          IF .NEWLENGTH LEQ 0 THEN RETURN;
;3069     3176  3
;3070     3177  3
;3071     3178  3          ! Output the new text of the line.  Here we scan the line's rendition
;3072     3179  3          ! codes and force a separate output operation each time the rendition
;3073     3180  3          ! codes of the text change.
;3074     3181  3          !
;3075     3182  3          OUTSTART = 0;
;3076     3183  3          OUTEND = 0;
;3077     3184  3          REND = .NEWREND[0];
;3078     3185  3          WHILE TRUE DO
;3079     3186  4              BEGIN
;3080     3187  5              IF (.OUTEND GEQ .NEWLENGTH) OR (.NEWREND[.OUTEND] NEQ .REND)
;3081     3188  4              THEN
;3082     3189  5                  BEGIN
;3083     3190  5                  TEXT_DESC[DSC$W_LENGTH] = .OUTEND - .OUTSTART;
;3084     3191  5                  TEXT_DESC[DSC$A_POINTER] = NEWTEXT[.OUTSTART];
;3085     3192  5                  STATUS = SCR$PUT_SCREEN(TEXT_DESC, .ROW, .OUTSTART + 1, .REND);
;3086     3193  5                  IF NOT .STATUS THEN SIGNAL(.STATUS);
;3087     3194  5                  IF .OUTEND GEQ .NEWLENGTH THEN EXITLOOP;
;3088     3195  5                  OUTSTART = .OUTEND;
;3089     3196  5                  REND = .NEWREND[.OUTSTART];
;3090     3197  4                  END;
;3091     3198  4
;3092     3199  4              OUTEND = .OUTEND + 1;
;3093     3200  3              END;
;3094     3201  3
;3095     3202  3          RETURN;
;3096     3203  2          END;
;3097     3204  2
;3098     3205  2
;3099     3206  2      ! The old (current) contents of the line on the screen are known.  Hence
;3100     3207  2      ! we initialize all indecies, counters, and flags needed for the minimal
;3101     3208  2      ! screen update algorithm.
;3102     3209  2      !
;3103     3210  2      OUTSTART = 0;
```

```
;  3104          3211  2     OUTEND = -1;
;  3105          3212  2     VALIDCOUNT = 100;
;  3106          3213  2     REND = .NEWREND[0];
;  3107          3214  2
;  3108          3215  2
;  3109          3216  2     ! Loop through the text vector to be output to the screen to identify the
;  3110          3217  2     ! sections of that text that must be updated on the screen.
;  3111          3218  2     !
;  3112          3219  2     INCR I FROM 0 TO .NEWLENGTH - 1 DO
;  3113          3220  3         BEGIN
;  3114          3221  3
;  3115          3222  3
;  3116          3223  3         ! If the current character matches what already is on the screen, we
;  3117          3224  3         ! increment the count of the number of consecutive valid characters
;  3118          3225  3         ! on the screen but do no output.  Note that if the current character
;  3119          3226  3         ! has a different rendition than the beginning of the character block,
;  3120          3227  3         ! we set VALIDCOUNT to 100 to prevent characters of different rendi-
;  3121          3228  3         ! tions from being included in the same output operation.
;  3122          3229  3         !
;  3123          3230  3         IF (.I LSS .OLDLENGTH)              AND
;  3124          3231  3            (.NEWTEXT[.I] EQL .OLDTEXT[.I]) AND
;  3125          3232  4            (.NEWREND[.I] EQL .OLDREND[.I])
;  3126          3233  3         THEN
;  3127          3234  4             BEGIN
;  3128          3235  4             IF .NEWREND[.I] NEQ .REND THEN VALIDCOUNT = 100;
;  3129          3236  4             VALIDCOUNT = .VALIDCOUNT + 1;
;  3130          3237  4             END
;  3131          3238  4
;  3132          3239  4
;  3133          3240  4         ! The current screen contents are invalid at this point.  Unless the
;  3134          3241  4         ! rendition is different or we just scanned past at least 5 consecutive
;  3135          3242  4         ! valid screen characters, we simply mark the current end position and
;  3136          3243  4         ! keep accumulating the text section to be output.  (Skipping over less
;  3137          3244  4         ! than 5 valid characters is not worth the cost of an extra cursor
;  3138          3245  4         ! positioning.)
;  3139          3246  4         !
;  3140          3247  4         ELSE IF (.NEWREND[.I] EQL .REND) AND (.VALIDCOUNT LSS 5)
;  3141          3248  3         THEN
;  3142          3249  4             BEGIN
;  3143          3250  4             OUTEND = .I;
;  3144          3251  4             VALIDCOUNT = 0;
;  3145          3252  4             END
;  3146          3253  4
;  3147          3254  4
;  3148          3255  4         ! We are at the start of a new invalid text position on the screen.
;  3149          3256  4         ! Now output the text section we just scanned past, as delimited by
;  3150          3257  4         ! the OUTSTART and OUTEND indecies.  The screen contents of that
;  3151          3258  4         ! section is invalid, except possibly for small embedded sections of
;  3152          3259  4         ! valid characters (less than 5 in a row per embedded section).
;  3153          3260  4         !
;  3154          3261  3         ELSE
;  3155          3262  4             BEGIN
;  3156          3263  4             IF .OUTEND GEQ .OUTSTART
;  3157          3264  4             THEN
;  3158          3265  5                 BEGIN
;  3159          3266  5                 TEXT_DESC[DSC$W_LENGTH] = .OUTEND - .OUTSTART + 1;
;  3160          3267  5                 TEXT_DESC[DSC$A_POINTER] = NEWTEXT[.OUTSTART];
```

```
;  3161     3268  5                STATUS = SCR$PUT_SCREEN(TEXT_DESC, .ROW, .OUTSTART + 1, .REND);
;  3162     3269  5                IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3163     3270  4                END;
;  3164     3271  4
;  3165     3272  4            OUTSTART = .I;
;  3166     3273  4            OUTEND = .I;
;  3167     3274  4            REND = .NEWREND[.I];
;  3168     3275  4            VALIDCOUNT = 0;
;  3169     3276  3            END;
;  3170     3277  3
;  3171     3278  2        END;                                ! End of character WHILE loop
;  3172     3279  2
;  3173     3280  2
;  3174     3281  2        ! Write out the last text section which is invalid on the screen if there
;  3175     3282  2        ! is such a section.
;  3176     3283  2        !
;  3177     3284  2        IF .OUTEND GEQ .OUTSTART
;  3178     3285  2        THEN
;  3179     3286  3            BEGIN
;  3180     3287  3            TEXT_DESC[DSC$W_LENGTH] = .OUTEND - .OUTSTART + 1;
;  3181     3288  3            TEXT_DESC[DSC$A_POINTER] = NEWTEXT[.OUTSTART];
;  3182     3289  3            STATUS = SCR$PUT_SCREEN(TEXT_DESC, .ROW, .OUTSTART + 1, .REND);
;  3183     3290  3            IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3184     3291  2            END;
;  3185     3292  2
;  3186     3293  2
;  3187     3294  2        ! Clear the tail end of the line if the new text for the line is shorter
;  3188     3295  2        ! than the old screen contents for this line.
;  3189     3296  2        !
;  3190     3297  2        IF .OLDLENGTH GTR .NEWLENGTH
;  3191     3298  2        THEN
;  3192     3299  3            BEGIN
;  3193     3300  3            STATUS = SCR$SET_CURSOR(.ROW, .NEWLENGTH + 1);
;  3194     3301  3            IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3195     3302  3            STATUS = SCR$ERASE_LINE();
;  3196     3303  3            IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3197     3304  2            END;
;  3198     3305  2
;  3199     3306  2
;  3200     3307  2        ! The line has been updated on the screen.  Now return.
;  3201     3308  2        !
;  3202     3309  2        RETURN;
;  3203     3310  2
;  3204     3311  1        END;


                                    0FFC 00000 DBG$SCR_OUTPUT_LINE_MINIMAL:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 3097
                5B 00000000G   00   9E 00002           MOVAB   SCR$ERASE_LINE, R11                 ;
                5A 00000000G   00   9E 00009           MOVAB   SCR$PUT_SCREEN, R10                 ;
                59 00000000G   00   9E 00010           MOVAB   LIB$SIGNAL, R9                      ;
                5E             08   C2 00017           SUBL2   #8, SP                              ;
          02    AE      010E   8F   B0 0001A           MOVW    #270, TEXT_DESC+2                   ; 3156
                          14   AC   D5 00020           TSTL    OLDTEXT                             ; 3163
```

```
                                                7B  12 00023          BNEQ    10$                                     ;
                                                01  DD 00025          PUSHL   #1                                      ; 3171
                                             1C AC  DD 00027          PUSHL   ROW                                     ;
                                 00000000G   00     02  FB 0002A     CALLS   #2, SCR$SET_CURSOR                      ;
                                             55     50  D0 00031     MOVL    R0, STATUS                              ;
                                             05     55  E8 00034     BLBS    STATUS, 1$                              ; 3172
                                                    55  DD 00037     PUSHL   STATUS                                  ;
                                             69     01  FB 00039     CALLS   #1, LIB$SIGNAL                          ;
                                             6B     00  FB 0003C 1$:  CALLS   #0, SCR$ERASE_LINE                      ; 3173
                                             55     50  D0 0003F     MOVL    R0, STATUS                              ;
                                             05     55  E8 00042     BLBS    STATUS, 2$                              ; 3174
                                                    55  DD 00045     PUSHL   STATUS                                  ;
                                             69     01  FB 00047     CALLS   #1, LIB$SIGNAL                          ;
                                          04 AC     D5  0004A 2$:     TSTL    NEWLENGTH                               ; 3175
                                                01  14  0004D       BGTR    3$                                      ;
                                                    04  0004F       RET                                             ;
                                                    54  D4 00050 3$:  CLRL    OUTSTART                                ; 3182
                                                    52  D4 00052     CLRL    OUTEND                                  ; 3183
                                             58  0C BC  9A 00054     MOVZBL  @NEWREND, REND                          ; 3184
                                                    53  D4 00058 4$:  CLRL    R3                                      ; 3187
                                       04    AC     52  D1 0005A     CMPL    OUTEND, NEWLENGTH                       ;
                                                04  19  0005E       BLSS    5$                                      ;
                                                    53  D6 00060     INCL    R3                                      ;
                                                09  11  00062       BRB     6$                                      ;
                58  0C BC42                       00  ED 00064 5$:  CMPZV   #0, #8, @NEWREND[OUTEND], REND          ;
                                                2F  13  0006B       BEQL    9$                                      ;
                                 6E             52  A3 0006D 6$:   SUBW3   OUTSTART, OUTEND, TEXT_DESC             ; 3190
                                       04    AE     08 BC44  9E 00071     MOVAB   @NEWTEXT[OUTSTART], TEXT_DESC+4     ; 3191
                                                    58  DD 00077     PUSHL   REND                                    ; 3192
                                                01  A4  9F 00079     PUSHAB  1(OUTSTART)                             ;
                                                1C  AC  DD 0007C     PUSHL   ROW                                     ;
                                                0C  AE  9F 0007F     PUSHAB  TEXT_DESC                               ;
                                             6A     04  FB 00082     CALLS   #4, SCR$PUT_SCREEN                      ;
                                             55     50  D0 00085     MOVL    R0, STATUS                              ;
                                             05     55  E8 00088     BLBS    STATUS, 7$                              ; 3193
                                                    55  DD 0008B     PUSHL   STATUS                                  ;
                                             69     01  FB 0008D     CALLS   #1, LIB$SIGNAL                          ;
                                             01     53  E9 00090 7$:  BLBC    R3, 8$                                  ; 3194
                                                    04  00093       RET                                             ;
                                             54     52  D0 00094 8$:  MOVL    OUTEND, OUTSTART                        ; 3195
                                             58  0C BC44  9A 00097     MOVZBL  @NEWREND[OUTSTART], REND                ; 3196
                                                    52  D6 0009C 9$:  INCL    OUTEND                                  ; 3199
                                                B8  11  0009E       BRB     4$                                      ; 3185
                                                    54  D4 000A0 10$: CLRL    OUTSTART                                ; 3210
                                             52     01  CE 000A2     MNEGL   #1, OUTEND                              ; 3211
                                             57  64 8F  9A 000A5     MOVZBL  #100, VALIDCOUNT                        ; 3212
                                             58  0C BC  9A 000A9     MOVZBL  @NEWREND, REND                          ; 3213
                                             53     01  CE 000AD     MNEGL   #1, I                                   ; 3230
                                                6F  11  000B0       BRB     17$                                     ;
                                 56             53  C1 000B2 11$:  ADDL3   NEWREND, I, R6                          ; 3235
                                       10    AC     53  D1 000B7     CMPL    I, OLDLENGTH                            ; 3230
                                                21  18  000BB       BGEQ    13$                                     ;
                                    14 BC43         08 BC43  91 000BD     CMPB    @NEWTEXT[I], @OLDTEXT[I]            ; 3231
                                                18  12  000C4       BNEQ    13$                                     ;
                                    18 BC43         0C BC43  91 000C6     CMPB    @NEWREND[I], @OLDREND[I]            ; 3232
                                                0F  12  000CD       BNEQ    13$                                     ;
                58                  66             00  ED 000CF     CMPZV   #0, #8, (R6), REND                      ; 3235
```

```
                                                04  13 000D4          BEQL    12$                                             ;
                                        57      64  8F 9A 000D6          MOVZBL  #100, VALIDCOUNT                        ;
                                                57  D6 000DA 12$:       INCL    VALIDCOUNT                              ; 3236
                                                43  11 000DC          BRB     17$                                     ; 3230
                58              66              08  00 ED 000DE 13$:       CMPZV   #0, #8, (R6), REND                      ; 3247
                                                05  12 000E3          BNEQ    14$                                     ;
                                        05      57  D1 000E5          CMPL    VALIDCOUNT, #5                          ;
                                                32  19 000E8          BLSS    16$                                     ;
                                        54      52  D1 000EA 14$:       CMPL    OUTEND, OUTSTART                        ; 3263
                                                27  19 000ED          BLSS    15$                                     ;
                                50      52          54  C3 000EF          SUBL3   OUTSTART, OUTEND, R0                    ; 3266
                                6E      50          01  A1 000F3          ADDW3   #1, R0, TEXT_DESC                       ;
                                    04  AE      08 BC44  9E 000F7          MOVAB   @NEWTEXT[OUTSTART], TEXT_DESC+4         ; 3267
                                                58  DD 000FD          PUSHL   REND                                    ; 3268
                                                01  A4 9F 000FF          PUSHAB  1(OUTSTART)                             ;
                                                1C  AC DD 00102          PUSHL   ROW                                     ;
                                                0C  AE 9F 00105          PUSHAB  TEXT_DESC                               ;
                                        6A          04  FB 00108          CALLS   #4, SCR$PUT_SCREEN                      ;
                                        55          50  D0 0010B          MOVL    R0, STATUS                              ;
                                        05          55  E8 0010E          BLBS    STATUS, 15$                             ; 3269
                                                    55  DD 00111          PUSHL   STATUS                                  ;
                                        69          01  FB 00113          CALLS   #1, LIB$SIGNAL                          ;
                                        54          53  D0 00116 15$:       MOVL    I, OUTSTART                             ; 3272
                                        58          66  9A 00119          MOVZBL  (R6), REND                              ; 3274
                                        52          53  D0 0011C 16$:       MOVL    I, OUTEND                               ; 3250
                                                    57  D4 0011F          CLRL    VALIDCOUNT                              ; 3251
                                8C              53      04  AC F2 00121 17$:       AOBLSS  NEWLENGTH, I, 11$               ; 3219
                                                54  52  D1 00126          CMPL    OUTEND, OUTSTART                        ; 3284
                                                    26  19 00129          BLSS    18$                                     ;
                                        52          54  C2 0012B          SUBL2   OUTSTART, R2                            ; 3287
                                6E      52          01  A1 0012E          ADDW3   #1, R2, TEXT_DESC                       ;
                                    04  AE      08 BC44  9E 00132          MOVAB   @NEWTEXT[OUTSTART], TEXT_DESC+4         ; 3288
                                                58  DD 00138          PUSHL   REND                                    ; 3289
                                                01  A4 9F 0013A          PUSHAB  1(OUTSTART)                             ;
                                                1C  AC DD 0013D          PUSHL   ROW                                     ;
                                                0C  AE 9F 00140          PUSHAB  TEXT_DESC                               ;
                                        6A          04  FB 00143          CALLS   #4, SCR$PUT_SCREEN                      ;
                                        55          50  D0 00146          MOVL    R0, STATUS                              ;
                                        05          55  E8 00149          BLBS    STATUS, 18$                             ; 3290
                                                    55  DD 0014C          PUSHL   STATUS                                  ;
                                        69          01  FB 0014E          CALLS   #1, LIB$SIGNAL                          ;
                                    04  AC      10      AC  D1 00151 18$:       CMPL    OLDLENGTH, NEWLENGTH                    ; 3297
                                                    28  15 00156          BLEQ    20$                                     ;
                7E              04  AC              01  C1 00158          ADDL3   #1, NEWLENGTH, -(SP)                    ; 3300
                                                1C  AC DD 0015D          PUSHL   ROW                                     ;
                    00000000G   00              02  FB 00160          CALLS   #2, SCR$SET_CURSOR                      ;
                                        55          50  D0 00167          MOVL    R0, STATUS                              ;
                                        05          55  E8 0016A          BLBS    STATUS, 19$                             ; 3301
                                                    55  DD 0016D          PUSHL   STATUS                                  ;
                                        69          01  FB 0016F          CALLS   #1, LIB$SIGNAL                          ;
                                        6B          00  FB 00172 19$:       CALLS   #0, SCR$ERASE_LINE                      ; 3302
                                        55          50  D0 00175          MOVL    R0, STATUS                              ;
                                        05          55  E8 00178          BLBS    STATUS, 20$                             ; 3303
                                                    55  DD 0017B          PUSHL   STATUS                                  ;
                                        69          01  FB 0017D          CALLS   #1, LIB$SIGNAL                          ;
                                                    04  00180 20$:       RET                                             ; 3311
```

; Routine Size: 385 bytes,   Routine Base: DBG$CODE + 1BB2

```
;  3206          3312  1 GLOBAL ROUTINE DBG$SCR_OUTPUT_SCREEN: NOVALUE =
;  3207          3313  1 !
;  3208          3314  1 ! FUNCTION
;  3209          3315  1 !       This routine outputs the current pasteboard contents to the terminal
;  3210          3316  1 !       screen.  It is normally called just before prompting for user input
;  3211          3317  1 !       so that the screen is updated to show the results or consequences of
;  3212          3318  1 !       the last DEBUG command the user entered.
;  3213          3319  1 !
;  3214          3320  1 ! INPUTS
;  3215          3321  1 !       NONE
;  3216          3322  1 !
;  3217          3323  1 ! OUTPUTS
;  3218          3324  1 !       NONE
;  3219          3325  1 !
;  3220          3326  1
;  3221          3327  2     BEGIN
;  3222          3328  2
;  3223          3329  2     LITERAL
;  3224          3330  2         WBUFSIZE = 256;                 ! Size in bytes of screen work buffer
;  3225          3331  2
;  3226          3332  2     LOCAL
;  3227          3333  2         BUFFER: VECTOR[132, BYTE],      ! Buffer for screen output strings
;  3228          3334  2         DCOL,                           ! Display column position for this line
;  3229          3335  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;  3230          3336  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to current Display Line Entry
;  3231          3337  2         END_LINE,                       ! Index of last line of scroll region
;  3232          3338  2         LAST_USED_LINE,                 ! Index of last used line on screen
;  3233          3339  2         N,                              ! Temporary index into output buffer
;  3234          3340  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to display name in ASCIC
;  3235          3341  2         OLDTEXT,                        ! Pointer to old text for line or zero
;  3236          3342  2         REGSIZE,                        ! Scrolling region size in lines
;  3237          3343  2         RENDITION_LINE,                 ! Flag set for lines with rendition vector
;  3238          3344  2         RENDPTR: REF VECTOR[,BYTE],     ! Pointer to line's rendition vector
;  3239          3345  2         REND_VECTOR: VECTOR[132,BYTE],  ! Temporary vector of rendition codes
;  3240          3346  2         R_INDEX,                        ! Rendition vector index (current index)
;  3241          3347  2         R_START,                        ! Rendition vector index (start index)
;  3242          3348  2         SCROLL_AMOUNT,                  ! Scrolling amount in lines (abs value)
;  3243          3349  2         SCROLL_DOWN,                    ! Flag set if scroll direction is down
;  3244          3350  2         START_LINE,                     ! Index of first line of scroll region
;  3245          3351  2         STATUS,                         ! Status code returned by screen routines
;  3246          3352  2         TEXTCNT,                        ! Number of characters in current line
;  3247          3353  2         TEXT_DESCR: BLOCK[8,BYTE],      ! String descriptor for screen text line
;  3248          3354  2         TEXTPTR: REF VECTOR[,BYTE],     ! Pointer to current line's ASCII text
;  3249          3355  2         WORK_BUF: VECTOR[WBUFSIZE,BYTE],! Screen buffer mode work buffer
;  3250          3356  2         WORK_BUF_DESCR: BLOCK[8,BYTE];  ! String descriptor for work buffer
;  3251          3357  2
;  3252          3358  2
;  3253          3359  2
;  3254          3360  2     ! Update the Pasteboard with the current contents and relative positions
;  3255          3361  2     ! of all the displays on the Screen Display List.
;  3256          3362  2     !
;  3257          3363  2     DBG$SCR_UPDATE_PASTEBOARD();
;  3258          3364  2
;  3259          3365  2
;  3260          3366  2     ! Set up screen buffering mode to optimize the screen output.
;  3261          3367  2     !
;  3262          3368  2     WORK_BUF_DESCR[DSC$B_CLASS] = DSC$K_CLASS_S;
```

```
;  3263          3369  2     WORK_BUF_DESCR[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  3264          3370  2     WORK_BUF_DESCR[DSC$W_LENGTH] = WBUFSIZE;
;  3265          3371  2     WORK_BUF_DESCR[DSC$A_POINTER] = WORK_BUF;
;  3266          3372  2     STATUS = SCR$SET_BUFFER(WORK_BUF_DESCR);
;  3267          3373  2     IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3268          3374  2
;  3269          3375  2
;  3270          3376  2     ! Set the scrolling region to be the whole screen.  This is only necessary
;  3271          3377  2     ! for VT100 terminals in which "Origin Mode" is set, meaning that all cursor
;  3272          3378  2     ! positioning is relative to the current scrolling region.  The normal state
;  3273          3379  2     ! is that cursor positioning is relative to the full physical screen, which
;  3274          3380  2     ! is what we require for our screen update to work correctly.  By setting
;  3275          3381  2     ! the scrolling region to be the whole screen, we make sure that the cursor
;  3276          3382  2     ! positioning commands below work as intended regardless of the setting of
;  3277          3383  2     ! the Origin Mode bit in the VT100.
;  3278          3384  2     !
;  3279          3385  2     STATUS = SCR$SET_SCROLL(1, 24);
;  3280          3386  2     IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3281          3387  2
;  3282          3388  2
;  3283          3389  2     ! Initialize START_LINE and END_LINE to suppress an erroneous error message
;  3284          3390  2     ! by the BLISS compiler about them not being initialized before use.  (They
;  3285          3391  2     ! are in fact initialized before use even without these two assignments.)
;  3286          3392  2     ! If the BLISS compiler is ever fixed, these two lines can be removed.
;  3287          3393  2     !
;  3288          3394  2     START_LINE = 0;
;  3289          3395  2     END_LINE = 0;
;  3290          3396  2
;  3291          3397  2
;  3292          3398  2     ! Loop through the lines of the pasteboard to discover all regions that can
;  3293          3399  2     ! best be output by scrolling regions of the terminal screen.
;  3294          3400  2     !
;  3295          3401  2     REGSIZE = 0;
;  3296          3402  2     INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  3297          3403  3         BEGIN
;  3298          3404  3
;  3299          3405  3
;  3300          3406  3         ! If this line is a regular text line, mark the start (if this is the
;  3301          3407  3         ! start) and end of the region of contiguous lines from the same dis-
;  3302          3408  3         ! play and increment the region size.
;  3303          3409  3         !
;  3304          3410  3         IF .PASTEBOARD[.I, DBG$B_PASTE_KIND] EQL DBG$K_PASTE_TEXT
;  3305          3411  3         THEN
;  3306          3412  4             BEGIN
;  3307          3413  4             DISPTR = .PASTEBOARD[.I, DBG$L_PASTE_DISPID];
;  3308          3414  4             IF .REGSIZE EQL 0 THEN START_LINE = .I;
;  3309          3415  4             END_LINE = .I;
;  3310          3416  4             REGSIZE = .REGSIZE + 1;
;  3311          3417  4             END
;  3312          3418  4
;  3313          3419  4
;  3314          3420  4         ! If the next line is not a regular text line, the potential scrolling
;  3315          3421  4         ! region has ended and we see if scrolling a screen region is the best
;  3316          3422  4         ! action.  If it is, we then do so.  Note that scrolling is disabled
;  3317          3423  4         ! if this is not a VT100 class terminal.
;  3318          3424  4         !
;  3319          3425  3         ELSE IF (.REGSIZE NEQ 0) AND .DBG$GL_VT100_FLAG
```

```
;  3320     3426  3          THEN
;  3321     3427  4              BEGIN
;  3322     3428  4
;  3323     3429  4
;  3324     3430  4              ! Determine the scrolling direction and amount from the Pasteboard
;  3325     3431  4              ! Entry for the start line of this screen region.
;  3326     3432  4              !
;  3327     3433  4              SCROLL_DOWN = TRUE;
;  3328     3434  4              SCROLL_AMOUNT = .PASTEBOARD[.START_LINE, DBG$W_PASTE_SCROLL];
;  3329     3435  4              IF .SCROLL_AMOUNT LSS 0
;  3330     3436  4              THEN
;  3331     3437  5                  BEGIN
;  3332     3438  5                  SCROLL_DOWN = FALSE;
;  3333     3439  5                  SCROLL_AMOUNT = - .SCROLL_AMOUNT;
;  3334     3440  4                  END;
;  3335     3441  4
;  3336     3442  4
;  3337     3443  4              ! If scrolling can be done on the region (the set of lines) we just
;  3338     3444  4              ! scanned, set up a scrolling region and then do the actual scroll-
;  3339     3445  4              ! ing in the appropriate direction.
;  3340     3446  4              !
;  3341     3447  5              IF (.SCROLL_AMOUNT NEQ 0) AND (.SCROLL_AMOUNT LSS .REGSIZE)
;  3342     3448  4              THEN
;  3343     3449  5                  BEGIN
;  3344     3450  5                  STATUS = SCR$SET_SCROLL(.START_LINE + 1, .END_LINE + 1);
;  3345     3451  5                  IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3346     3452  5
;  3347     3453  5
;  3348     3454  5                  ! If the scrolling direction is down, meaning that the window
;  3349     3455  5                  ! moves down the display text (positive scrolling amount),
;  3350     3456  5                  ! shift the screen contents up the specified number of lines.
;  3351     3457  5                  !
;  3352     3458  5                  IF .SCROLL_DOWN
;  3353     3459  5                  THEN
;  3354     3460  6                      BEGIN
;  3355     3461  6
;  3356     3462  6
;  3357     3463  6                      ! Start by updating the old screen image by moving all
;  3358     3464  6                      ! lines up that we are about to scroll.
;  3359     3465  6                      !
;  3360     3466  6                      INCR J FROM .START_LINE + .SCROLL_AMOUNT TO .END_LINE DO
;  3361     3467  7                          BEGIN
;  3362     3468  7                          OLD_CNT[.J - .SCROLL_AMOUNT] = .OLD_CNT[.J];
;  3363     3469  7                          OLD_REND[.J - .SCROLL_AMOUNT] = .OLD_REND[.J];
;  3364     3470  7                          OLD_VALID[.J - .SCROLL_AMOUNT] = .OLD_VALID[.J];
;  3365     3471  7                          CH$MOVE(.OLD_CNT[.J], OLD_SCREEN[.J*132],
;  3366     3472  7                                  OLD_SCREEN[(.J - .SCROLL_AMOUNT)*132]);
;  3367     3473  7                          CH$MOVE(.OLD_CNT[.J], OLD_SCREEN_REND[.J*132],
;  3368     3474  7                                  OLD_SCREEN_REND[(.J - .SCROLL_AMOUNT)*132]);
;  3369     3475  6                          END;
;  3370     3476  6
;  3371     3477  6
;  3372     3478  6                      ! Then do the actual scrolling on the screen.  Note that we
;  3373     3479  6                      ! only shift up existing lines on the screen here.  The
;  3374     3480  6                      ! new lines to be scrolled in are marked as invalid here so
;  3375     3481  6                      ! that they get written during the screen update below.
;  3376     3482  6                      !
```

```
;  3377     3483  6                STATUS = SCR$SET_CURSOR(.END_LINE + 1, 1);
;  3378     3484  6                IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3379     3485  6                INCR J FROM 1 TO .SCROLL_AMOUNT DO
;  3380     3486  7                    BEGIN
;  3381     3487  7                    OLD_VALID[.END_LINE - .J + 1] = FALSE;
;  3382     3488  7                    STATUS = SCR$UP_SCROLL();
;  3383     3489  7                    IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3384     3490  6                    END;
;  3385     3491  6
;  3386     3492  6                END
;  3387     3493  6
;  3388     3494  6
;  3389     3495  6            ! Otherwise, scroll the region up by SCROLL_AMOUNT lines.  This
;  3390     3496  6            ! means that we move toward the top of the display and shift
;  3391     3497  6            ! the corresponding lines on the screen down the same amount.
;  3392     3498  6            !
;  3393     3499  5            ELSE
;  3394     3500  6                BEGIN
;  3395     3501  6
;  3396     3502  6
;  3397     3503  6                ! Start by updating the old screen image by moving all
;  3398     3504  6                ! lines down that we are about to scroll.
;  3399     3505  6                !
;  3400     3506  6                DECR J FROM .END_LINE TO .START_LINE + .SCROLL_AMOUNT DO
;  3401     3507  7                    BEGIN
;  3402     3508  7                    OLD_CNT[.J] = .OLD_CNT[.J - .SCROLL_AMOUNT];
;  3403     3509  7                    OLD_REND[.J] = .OLD_REND[.J - .SCROLL_AMOUNT];
;  3404     3510  7                    OLD_VALID[.J] = .OLD_VALID[.J - .SCROLL_AMOUNT];
;  3405     3511  7                    CH$MOVE(.OLD_CNT[.J],
;  3406     3512  7                            OLD_SCREEN[(.J - .SCROLL_AMOUNT)*132],
;  3407     3513  7                            OLD_SCREEN[.J*132]);
;  3408     3514  7                    CH$MOVE(.OLD_CNT[.J],
;  3409     3515  7                            OLD_SCREEN_REND[(.J - .SCROLL_AMOUNT)*132],
;  3410     3516  7                            OLD_SCREEN_REND[.J*132]);
;  3411     3517  6                    END;
;  3412     3518  6
;  3413     3519  6
;  3414     3520  6                ! Then do the actual scrolling on the screen.  Note that we
;  3415     3521  6                ! only shift down existing lines on the screen here.  The
;  3416     3522  6                ! new lines to be scrolled in are marked as invalid here so
;  3417     3523  6                ! that they get written during the screen update below.
;  3418     3524  6                !
;  3419     3525  6                STATUS = SCR$SET_CURSOR(.START_LINE + 1, 1);
;  3420     3526  6                IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3421     3527  6                INCR J FROM 1 TO .SCROLL_AMOUNT DO
;  3422     3528  7                    BEGIN
;  3423     3529  7                    OLD_VALID[.START_LINE + .J - 1] = FALSE;
;  3424     3530  7                    STATUS = SCR$DOWN_SCROLL();
;  3425     3531  7                    IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3426     3532  6                    END;
;  3427     3533  6
;  3428     3534  5                END;
;  3429     3535  5
;  3430     3536  4            END;
;  3431     3537  4
;  3432     3538  4
;  3433     3539  4        ! Set the region size back to zero for any non-text line.
```

```
;  3434     3540  4              !
;  3435     3541  4              REGSIZE = 0;
;  3436     3542  3              END;
;  3437     3543  3
;  3438     3544  2          END;                                ! End of loop to find scrolling regions
;  3439     3545  2
;  3440     3546  2
;  3441     3547  2          ! If this is not a VT100 class terminal (for example, if it is a VT52),
;  3442     3548  2          ! invalidate all lines in the old screen image.  These lines are no good
;  3443     3549  2          ! since the whole screen always scrolls when the user enters input.
;  3444     3550  2          !
;  3445     3551  2          IF NOT .DBG$GL_VT100_FLAG
;  3446     3552  2          THEN
;  3447     3553  3              BEGIN
;  3448     3554  3              INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  3449     3555  3                  OLD_VALID[.I] = FALSE;
;  3450     3556  3
;  3451     3557  2              END;
;  3452     3558  2
;  3453     3559  2
;  3454     3560  2          ! Determine which is the last used line in the pasteboard.  Any null lines
;  3455     3561  2          ! before this line (i.e., above at least one display) will be blanked out
;  3456     3562  2          ! instead of being left alone as they are in the user program scrolling
;  3457     3563  2          ! region.
;  3458     3564  2          !
;  3459     3565  2          DECR I FROM DBG$K_PASTE_SIZE - 1 TO 0 DO
;  3460     3566  3              BEGIN
;  3461     3567  3              LAST_USED_LINE = .I;
;  3462     3568  3              IF .PASTEBOARD[.I, DBG$B_PASTE_KIND] NEQ DBG$K_PASTE_NULL THEN EXITLOOP;
;  3463     3569  2              END;
;  3464     3570  2
;  3465     3571  2
;  3466     3572  2          ! Loop through the lines in the pasteboard and output each such line to the
;  3467     3573  2          ! terminal screen.  Note that we apply the necessary shift-count to each
;  3468     3574  2          ! output line so that lines are correctly scrolled left or right.  Each
;  3469     3575  2          ! line is written out by the OUTPUT_LINE_MINIMAL routine which applies a
;  3470     3576  2          ! minimal screen update algorithm to update the screen with the new text.
;  3471     3577  2          !
;  3472     3578  2          INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  3473     3579  3              BEGIN
;  3474     3580  3              RENDITION_LINE = FALSE;
;  3475     3581  3
;  3476     3582  3
;  3477     3583  3              ! Construct the line to be output to the screen based on the line's
;  3478     3584  3              ! kind.
;  3479     3585  3              !
;  3480     3586  3              CASE .PASTEBOARD[.I, DBG$B_PASTE_KIND]
;  3481     3587  3                                  FROM DBG$K_PASTE_NULL TO DBG$K_PASTE_BORDER OF
;  3482     3588  3                  SET
;  3483     3589  3
;  3484     3590  3
;  3485     3591  3                  ! Handle the Null line.  A null line before the last line used for
;  3486     3592  3                  ! part of a display is blanked out.  A null line after the last
;  3487     3593  3                  ! used line is left alone since it is part of the user program
;  3488     3594  3                  ! scrolling region on the terminal screen.
;  3489     3595  3                  !
;  3490     3596  3                  [DBG$K_PASTE_NULL]:
```

```
;  3491     3597  4                BEGIN
;  3492     3598  4                TEXTPTR = BUFFER;
;  3493     3599  4                TEXTCNT = 0;
;  3494     3600  4                IF .I GTR .LAST_USED_LINE THEN TEXTPTR = 0;
;  3495     3601  4                END;
;  3496     3602  3
;  3497     3603  3
;  3498     3604  3            ! Handle an ordinary text line from some display.  Here we compute
;  3499     3605  3            ! the text start address and length by taking the column shift
;  3500     3606  3            ! count (DCOL) into account so that display lines can be correctly
;  3501     3607  3            ! scrolled left or right.  (Scrolling right means that the window
;  3502     3608  3            ! moves to the right over the display text--the actual screen con-
;  3503     3609  3            ! tents are then shifted to the left--and vice versa for scrolling
;  3504     3610  3            ! left.)  Note that lines from source displays are formatted by a
;  3505     3611  3            ! separate routine.  Also note that we check for lines with per-
;  3506     3612  3            ! character rendition information.
;  3507     3613  3            !
;  3508     3614  3            [DBG$K_PASTE_TEXT]:
;  3509     3615  4                BEGIN
;  3510     3616  4                DLEPTR = .PASTEBOARD[.I, DBG$L_PASTE_DLINE];
;  3511     3617  4                IF .DLEPTR[DBG$V_DLINE_SOURCEF[G]
;  3512     3618  4                THEN
;  3513     3619  4                    FORMAT_SOURCE_LINE(.DISPTR, .DLEPTR,
;  3514     3620  4                                       TEXTPTR, TEXTCNT, BUFFER)
;  3515     3621  4
;  3516     3622  4                ELSE
;  3517     3623  5                    BEGIN
;  3518     3624  5                    DCOL = .DISPTR[DBG$W_DISP_DCOL];
;  3519     3625  5                    TEXTPTR = DLEPTR[DBG$A_DLINE_TEXT];
;  3520     3626  5                    IF .DLEPTR[DBG$V_DLINE_RENDF[G]
;  3521     3627  5                    THEN
;  3522     3628  6                        BEGIN
;  3523     3629  6                        RENDITION_LINE = TRUE;
;  3524     3630  6                        RENDPTR = TEXTPTR[.TEXTPTR[0] + .DCOL];
;  3525     3631  5                        END;
;  3526     3632  5
;  3527     3633  5                    TEXTCNT = MIN(.DBG$SRC_TERM_WIDTH,
;  3528     3634  5                                  MAX(0, .TEXTPTR[0] - .DCOL + 1));
;  3529     3635  5                    IF .TEXTCNT NEQ 0 THEN TEXTPTR = TEXTPTR[.DCOL];
;  3530     3636  4                    END;
;  3531     3637  4
;  3532     3638  3                END;
;  3533     3639  3
;  3534     3640  3
;  3535     3641  3            ! Handle a blank line which is part of a display's text.
;  3536     3642  3            !
;  3537     3643  3            [DBG$K_PASTE_BLANK]:
;  3538     3644  4                BEGIN
;  3539     3645  4                TEXTPTR = BUFFER;
;  3540     3646  4                TEXTCNT = 0;
;  3541     3647  3                END;
;  3542     3648  3
;  3543     3649  3
;  3544     3650  3            ! Handle a label line on top of a display.  This line consists of a
;  3545     3651  3            ! line of dashes into which the display name has been filled.  Note
;  3546     3652  3            ! that we also mark the label line accordingly if this display is
;  3547     3653  3            ! currently selected for input, output, scrolling, or source.
```

```
;  3548          3654  3          !
;  3549          3655  3          [DBG$K_PASTE_LABEL]:
;  3550          3656  4              BEGIN
;  3551          3657  4
;  3552          3658  4
;  3553          3659  4              ! Fill in the name of the display in the label line.
;  3554          3660  4              !
;  3555          3661  4              TEXTPTR = BUFFER;
;  3556          3662  4              TEXTCNT = .DBG$SRC_TERM_WIDTH;
;  3557          3663  4              CH$FILL('-', .TEXTCNT, BUFFER);
;  3558          3664  4              DISPTR = .PASTEBOARD[.I, DBG$L_PASTE_DISPID];
;  3559          3665  4              NAMEPTR = DISPTR[DBG$A_DISP_NAME];
;  3560          3666  4              CH$MOVE(MIN(80, .NAMEPTR[0]), NAMEPTR[1], BUFFER[2]);
;  3561          3667  4              N = MIN(80, .NAMEPTR[0]) + 2;
;  3562          3668  4
;  3563          3669  4
;  3564          3670  4              ! If this is a source line display, fill in the module name of
;  3565          3671  4              ! the source text.  For example: "-DISPNAM: module MODNAM-".
;  3566          3672  4              !
;  3567          3673  4              IF (.DISPTR[DBG$B_DISP_KIND] EQL DBG$K_DISP_SOURCE) AND
;  3568          3674  5                 (.DISPTR[DBG$L_DISP_MODPTR] NEQ 0)
;  3569          3675  4              THEN
;  3570          3676  5                  BEGIN
;  3571          3677  5                  CH$MOVE(9, UPLIT BYTE(%ASCII ': module '), BUFFER[.N]);
;  3572          3678  5                  N = .N + 9;
;  3573          3679  5                  DBG$STA_SYMNAME(.DISPTR[DBG$L_DISP_MODPTR], NAMEPTR);
;  3574          3680  5                  CH$MOVE(MIN(32, .NAMEPTR[0]), NAMEPTR[1], BUFFER[.N]);
;  3575          3681  5                  N = .N + MIN(32, .NAMEPTR[0]);
;  3576          3682  4                  END;
;  3577          3683  4
;  3578          3684  4
;  3579          3685  4              ! Fill in the display selections that apply to this display.
;  3580          3686  4              ! This can be -input-, -output-, -source-, and -scroll-.
;  3581          3687  4              !
;  3582          3688  4              N = MIN(50, .N + 2);
;  3583          3689  4              IF .DISPTR EQL .DBG$SCR_CURDISP_INPUT
;  3584          3690  4              THEN
;  3585          3691  5                  BEGIN
;  3586          3692  5                  CH$MOVE(7, UPLIT BYTE(%ASCII '-input-'), BUFFER[.N]);
;  3587          3693  5                  N = .N + 6;
;  3588          3694  4                  END;
;  3589          3695  4
;  3590          3696  4              IF .DISPTR EQL .DBG$SCR_CURDISP_OUTPUT
;  3591          3697  4              THEN
;  3592          3698  5                  BEGIN
;  3593          3699  5                  CH$MOVE(8, UPLIT BYTE(%ASCII '-output-'), BUFFER[.N]);
;  3594          3700  5                  N = .N + 7;
;  3595          3701  4                  END;
;  3596          3702  4
;  3597          3703  4              IF .DISPTR EQL .DBG$SCR_CURDISP_SOURCE
;  3598          3704  4              THEN
;  3599          3705  5                  BEGIN
;  3600          3706  5                  CH$MOVE(8, UPLIT BYTE(%ASCII '-source-'), BUFFER[.N]);
;  3601          3707  5                  N = .N + 7;
;  3602          3708  4                  END;
;  3603          3709  4
;  3604          3710  4              IF .DISPTR EQL .DBG$SCR_CURDISP_SCROLL
```

```
;  3605     3711  4                THEN
;  3606     3712  5                    BEGIN
;  3607     3713  5                    CH$MOVE(8, UPLIT BYTE(%ASCII '-scroll-'), BUFFER[.N]);
;  3608     3714  5                    N = .N + 7;
;  3609     3715  4                    END;
;  3610     3716  4
;  3611     3717  3                END;
;  3612     3718  3
;  3613     3719  3
;  3614     3720  3            ! Handle the border line below the lowest display on the screen.
;  3615     3721  3            ! This line consists of nothing but a line of dashes.
;  3616     3722  3            !
;  3617     3723  3            [DBG$K_PASTE_BORDER]:
;  3618     3724  4                BEGIN
;  3619     3725  4                TEXTPTR = BUFFER;
;  3620     3726  4                TEXTCNT = .DBG$SRC_TERM_WIDTH;
;  3621     3727  4                CH$FILL('-', .TEXTCNT, BUFFER);
;  3622     3728  3                END;
;  3623     3729  3
;  3624     3730  3
;  3625     3731  3            ! Any other kind value is an internal DEBUG error.
;  3626     3732  3            !
;  3627     3733  3            [INRANGE, OUTRANGE]:
;  3628     3734  3                $DBG_ERROR('DBGSCREEN\OUTPUT_SCREEN');
;  3629     3735  3
;  3630     3736  3            TES;
;  3631     3737  3
;  3632     3738  3
;  3633     3739  3            ! If the line of text is a "null" line, i.e. part of the user program
;  3634     3740  3            ! scrolling region or otherwise outside any screen display, simply
;  3635     3741  3            ! mark the line contents as "invalid" and do not output any text.
;  3636     3742  3            !
;  3637     3743  3            IF .TEXTPTR EQL 0
;  3638     3744  3            THEN
;  3639     3745  3                OLD_VALID[.I] = FALSE
;  3640     3746  3
;  3641     3747  3
;  3642     3748  3            ! If this is a normal text line, we output that text with the appropri-
;  3643     3749  3            ! ate renditions using a minimal screen update algorithm.
;  3644     3750  3            !
;  3645     3751  3            ELSE
;  3646     3752  4                BEGIN
;  3647     3753  4
;  3648     3754  4
;  3649     3755  4                ! If the new line does not have a rendition vector (i.e., if it has
;  3650     3756  4                ! a single rendition for the whole line), create a rendition vector
;  3651     3757  4                ! for it.
;  3652     3758  4                !
;  3653     3759  4                IF NOT .RENDITION_LINE
;  3654     3760  4                THEN
;  3655     3761  5                    BEGIN
;  3656     3762  5                    RENDPTR = REND_VECTOR[0];
;  3657     3763  5                    CH$FILL(.PASTEBOARD[.I, DBG$B_PASTE_REND], .TEXTCNT, .RENDPTR);
;  3658     3764  4                    END;
;  3659     3765  4
;  3660     3766  4
;  3661     3767  4                ! Update the screen with the new text of this line.
```

```
;  3662          3768  4              !
;  3663          3769  4              OLDTEXT = 0;
;  3664          3770  4              IF .OLD_VALID[.I] THEN OLDTEXT = OLD_SCREEN[.I*132];
;  3665          3771  4              DBG$SCR_OUTPUT_LINE_MINIMAL(.TEXTCNT, .TEXTPTR, .RENDPTR,
;  3666          3772  4                  .OLD_CNT[.I], .OLDTEXT, OLD_SCREEN_REND[.I*132], .I + 1);
;  3667          3773  4
;  3668          3774  4
;  3669          3775  4              ! Update the old screen image to reflect the new screen contents.
;  3670          3776  4              !
;  3671          3777  4              CH$MOVE(.TEXTCNT, .TEXTPTR, OLD_SCREEN[.I*132]);
;  3672          3778  4              CH$MOVE(.TEXTCNT, .RENDPTR, OLD_SCREEN_REND[.I*132]);
;  3673          3779  4              OLD_CNT[.I] = .TEXTCNT;
;  3674          3780  4              OLD_REND[.I] = .PASTEBOARD[.I, DBG$B_PASTE_REND];
;  3675          3781  4              IF .RENDITION_LINE THEN OLD_REND[.I] = 255;
;  3676          3782  4              OLD_VALID[.I] = TRUE;
;  3677          3783  3              END;
;  3678          3784  3
;  3679          3785  2          END;                                ! End of loop over pasteboard lines
;  3680          3786  2
;  3681          3787  2
;  3682          3788  2      ! Set up a scrolling region on the screen which consists of those lines at
;  3683          3789  2      ! the bottom of the screen which are not covered by any display. We dynami-
;  3684          3790  2      ! cally determine how many such lines there are.
;  3685          3791  2      !
;  3686          3792  2      START_LINE = DBG$K_PASTE_SIZE + 1;
;  3687          3793  2      END_LINE = 24;
;  3688          3794  2      DECR I FROM DBG$K_PASTE_SIZE TO 1 DO
;  3689          3795  3          BEGIN
;  3690          3796  3          IF .PASTEBOARD[.I - 1, DBG$B_PASTE_KIND] NEQ DBG$K_PASTE_NULL
;  3691          3797  3          THEN
;  3692          3798  3              EXITLOOP;
;  3693          3799  3
;  3694          3800  3          START_LINE = .I;
;  3695          3801  2          END;
;  3696          3802  2
;  3697          3803  2      STATUS = SCR$SET_SCROLL(.START_LINE, .END_LINE);
;  3698          3804  2      IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3699          3805  2
;  3700          3806  2
;  3701          3807  2      ! Finally position the cursor at the start of the last line of the screen.
;  3702          3808  2      !
;  3703          3809  2      STATUS = SCR$SET_CURSOR(24, 1);
;  3704          3810  2      IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3705          3811  2
;  3706          3812  2
;  3707          3813  2      ! Output the buffer built up in screen buffer mode to force all output to
;  3708          3814  2      ! the terminal screen.  The screen update is then complete.
;  3709          3815  2      !
;  3710          3816  2      STATUS = SCR$PUT_BUFFER(0);
;  3711          3817  2      IF NOT .STATUS THEN SIGNAL(.STATUS);
;  3712          3818  2
;  3713          3819  2
;  3714          3820  2      ! If screen logging is turned on, we output the current screen contents to
;  3715          3821  2      ! the DEBUG log file.  After that, we return.
;  3716          3822  2      !
;  3717          3823  2      IF .DBG$GL_SCREEN_LOG THEN DBG$SCR_SCREEN_TO_LOGFILE();
;  3718          3824  2      RETURN;
```

```
; 3719          3825  2
; 3720          3826  1     END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                        20  65  6C  75  64  6F  6D  20  3A  0063F P.AHF:  .ASCII  \: module \
                            2D  74  75  70  6E  69  2D  00648 P.AHG:  .ASCII  \-input-\
                        2D  74  75  70  74  75  6F  2D  0064F P.AHH:  .ASCII  \-output-\
                        2D  65  63  72  75  6F  73  2D  00657 P.AHI:  .ASCII  \-source-\
                        2D  6C  6C  6F  72  63  73  2D  0065F P.AHJ:  .ASCII  \-scroll-\
50  54  55  4F  5C  4E  45  45  52  43  53  47  42  44  17  00667 P.AHK:  .ASCII  <23>\DBGSCREEN\<92>\OUTPUT_SCREEN\
                        4E  45  45  52  43  53  5F  54  55  00676

                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                    0FFC 00000          .ENTRY  DBG$SCR_OUTPUT_SCREEN, Save R2,R3,R4,R5,R6,-; 3312
                                                                                R7,R8,R9,R10,R11
                            5E      FDBC        CE  9E 00002            MOVAB   -580(SP), SP
                  0000V     CF                  00  FB 00007            CALLS   #0, DBG$SCR_UPDATE_PASTEBOARD                ; 3363
                     2C     AE  010E0100        8F  D0 0000C            MOVL    #17694976, WORK_BUF_DESCR                   ; 3370
                     30     AE          34      AE  9E 00014            MOVAB   WORK_BUF, WORK_BUF_DESCR+4                  ; 3371
                            AE          2C      AE  9F 00019            PUSHAB  WORK_BUF_DESCR                              ; 3372
          00000000G         00                  01  FB 0001C            CALLS   #1, SCR$SET_BUFFER
                            6E                  50  D0 00023            MOVL    R0, STATUS
                            09                  6E  E8 00026            BLBS    STATUS, 1$                                  ; 3373
                                                6E  DD 00029            PUSHL   STATUS
          00000000G         00                  01  FB 0002B            CALLS   #1, LIB$SIGNAL
                                                18  DD 00032 1$:        PUSHL   #24                                         ; 3385
                                                01  DD 00034            PUSHL   #1
          00000000G         00                  02  FB 00036            CALLS   #2, SCR$SET_SCROLL
                            6E                  50  D0 0003D            MOVL    R0, STATUS
                            09                  6E  E8 00040            BLBS    STATUS, 2$                                  ; 3386
                                                6E  DD 00043            PUSHL   STATUS
          00000000G         00                  01  FB 00045            CALLS   #1, LIB$SIGNAL
                                                58  7C 0004C 2$:        CLRQ    START_LINE                                  ; 3394
                                          14    AE  D4 0004E            CLRL    REGSIZE                                     ; 3401
                                                57  D4 00051            CLRL    I                                           ; 3402
50                      57                      0C  C5 00053 3$:        MULL3   #12, I, R0                                  ; 3410
                        02  00000000'EF40        91  00057            CMPB    PASTEBOARD[R0], #2
                                                1C  12 0005F            BNEQ    6$
                            00000000'EF40       9F  00061            PUSHAB  PASTEBOARD+4[R0]                            ; 3413
                  04    AE                      9E  D0 00068            MOVL    @(SP)+, DISPTR
                                          14    AE  D5 0006C            TSTL    REGSIZE                                     ; 3414
                                                03  12 0006F            BNEQ    4$
                            58                  57  D0 00071            MOVL    I, START_LINE
                            59                  57  D0 00074 4$:        MOVL    I, END_LINE                                 ; 3415
                                          14    AE  D6 00077            INCL    REGSIZE                                     ; 3416
                                             01E0   31 0007A 5$:        BRW     26$                                         ; 3410
                                          14    AE  D5 0007D 6$:        TSTL    REGSIZE                                     ; 3425
                                                F8  13 00080            BEQL    5$
                        F1  00000000'           EF  E9 00082            BLBC    DBG$GL_VT100_FLAG, 5$
                  1C    AE                      01  D0 00089            MOVL    #1, SCROLL_DOWN                             ; 3433
5A                      58                      0C  C5 0008D            MULL3   #12, START_LINE, R10                        ; 3434
```

```
                                00000000'EF4A  9F 00091          PUSHAB  PASTEBOARD+2[R10]
                          10 AE         9E  32 00098          CVTWL   @(SP)+, SCROLL_AMOUNT
                                        08  18 0009C          BGEQ    7$                              ;3435
                             1C AE      D4  0009E          CLRL    SCROLL_DOWN                     ;3438
                          10 AE   10 AE  CE  000A1          MNEGL   SCROLL_AMOUNT, SCROLL_AMOUNT    ;3439
                                        03  12 000A6 7$:    BNEQ    9$                              ;3447
                                     01AF  31 000A8 8$:    BRW     25$
                          14 AE   10 AE  D1  000AB 9$:    CMPL    SCROLL_AMOUNT, REGSIZE
                                        F6  18 000B0          BGEQ    8$
                             01 A9      9F  000B2          PUSHAB  1(END_LINE)                     ;3450
                             01 A8      9F  000B5          PUSHAB  1(START_LINE)
                00000000G    00         02  FB 000B8          CALLS   #2, SCR$SET_SCROLL
                             6E         50  D0 000BF          MOVL    R0, STATUS
                             09         6E  E8 000C2          BLBS    STATUS, 10$                     ;3451
                             6E         DD  000C5          PUSHL   STATUS
                00000000G    00         01  FB 000C7          CALLS   #1, LIB$SIGNAL
                          18 AE   10 BE48  9E  000CE 10$:    MOVAB   @SCROLL_AMOUNT[START_LINE], 24(SP)  ;3466
                             03      1C AE  E8  000D4          BLBS    SCROLL_DOWN, 11$                ;3458
                                     00C0  31 000D8          BRW     18$
                56        18 AE         01  C3 000DB 11$:    SUBL3   #1, 24(SP), J                   ;3466
                                        6A  11 000E0          BRB     13$
                5A           56      10 AE  C3 000E2 12$:    SUBL3   SCROLL_AMOUNT, J, R10           ;3468
                00000000'EF4A 00000000'EF46  D0 000E7          MOVL    OLD_CNT[J], OLD_CNT[R10]
                00000000'EF4A 00000000'EF46  90 000F4          MOVB    OLD_REND[J], OLD_REND[R10]      ;3469
        50 00000000'    EF      01      56  EF 00101          EXTZV   J, #1, OLD_VALID, R0            ;3470
00000000'   EF          01      5A      50  F0 0010A          INSV    R0, R10, #1, OLD_VALID
                5B           56 00000084  8F  C5 00113          MULL3   #132, J, R11                    ;3471
                             5A 00000084  8F  C4 00116          MULL2   #132, R10                       ;3472
                                00000000'EF46  DF 00122          PUSHAL  OLD_CNT[J]
00000000'EF4A 00000000'EF4B             9E  28 00129          MOVC3   @(SP)+, OLD_SCREEN[R11], OLD_SCREEN[R10]
                                00000000'EF46  DF 00137          PUSHAL  OLD_CNT[J]                      ;3474
00000000'EF4A 00000000'EF4B             9E  28 0013E          MOVC3   @(SP)+, OLD_SCREEN_REND[R11], -
                                                                        OLD_SCREEN_REND[R10]
             92           56         59  F3 0014C 13$:    AOBLEQ  END_LINE, J, 12$                ;3466
                                     01  DD 00150          PUSHL   #1                              ;3483
                                01   A9  9F 00152          PUSHAB  1(END_LINE)
                00000000G    00         02  FB 00155          CALLS   #2, SCR$SET_CURSOR
                             6E         50  D0 0015C          MOVL    R0, STATUS
                             09         6E  E8 0015F          BLBS    STATUS, 14$                     ;3484
                             6E         DD  00162          PUSHL   STATUS
                00000000G    00         01  FB 00164          CALLS   #1, LIB$SIGNAL
                                        52  D4 0016B 14$:    CLRL    J                               ;3485
                                        24  11 0016D          BRB     17$
             5B           59         52  C3 0016F 15$:    SUBL3   J, END_LINE, R11                ;3487
                                        5B  D6 00173          INCL    R11
             00 00000000'    EF      5B  E5 00175          BBCC    R11, OLD_VALID, 16$
                00000000G    00         00  FB 0017D 16$:    CALLS   #0, SCR$UP_SCROLL               ;3488
                             6E         50  D0 00184          MOVL    R0, STATUS
                             09         6E  E8 00187          BLBS    STATUS, 17$                     ;3489
                             6E         DD  0018A          PUSHL   STATUS
                00000000G    00         01  FB 0018C          CALLS   #1, LIB$SIGNAL
             D7           52      10 AE  F3 00193 17$:    AOBLEQ  SCROLL_AMOUNT, J, 15$           ;3485
                                  00BF  31 00198          BRW     25$                             ;3458
                             56         59  D0 0019B 18$:    MOVL    END_LINE, J                     ;3506
                                        6D  11 0019E          BRB     20$
             5A           56      10 AE  C3 001A0 19$:    SUBL3   SCROLL_AMOUNT, J, R10           ;3508
                00000000'EF46 00000000'EF4A  D0 001A5          MOVL    OLD_CNT[R10], OLD_CNT[J]
```

```
                                00000000'EF46 00000000'EF4A  90 001B2          MOVB    OLD_REND[R10], OLD_REND[J]                      ; 3509
          50 00000000'  EF                01            5A  EF 001BF          EXTZV   R10, #1, OLD_VALID, R0                          ; 3510
00000000'  EF                   01        56            50  F0 001C8          INSV    R0, J, #1, OLD_VALID                            ;
                                5B        5A 00000084   8F  C5 001D1          MULL3   #132, R10, R11                                  ; 3512
                                5A        56 00000084   8F  C5 001D9          MULL3   #132, J, R10                                    ; 3513
                                                00000000'EF46 DF 001E1          PUSHAL  OLD_CNT[J]                                      ;
                   00000000'EF4A 00000000'EF4B            9E 28 001E8          MOVC3   @(SP)+, OLD_SCREEN[R11], OLD_SCREEN[R10]        ;
                                                00000000'EF46 DF 001F6          PUSHAL  OLD_CNT[J]                                      ; 3516
                   00000000'EF4A 00000000'EF4B            9E 28 001FD          MOVC3   @(SP)+, OLD_SCREEN_REND[R11], -                 ;
                                                                                       OLD_SCREEN_REND[R10]                             ;
                                           56   D7 0020B          DECL    J                                                       ; 3506
                                     18    AE   56   D1 0020D 20$:  CMPL    J, 24(SP)                                               ;
                                           8D   18 00211          BGEQ    19$                                                     ;
                                           01   DD 00213          PUSHL   #1                                                      ; 3525
                                        01 A8   9F 00215          PUSHAB  1(START_LINE)                                           ;
                                00000000G  00   02 FB 00218          CALLS   #2, SCR$SET_CURSOR                                      ;
                                           6E   50 DO 0021F          MOVL    R0, STATUS                                              ;
                                           09   6E E8 00222          BLBS    STATUS, 21$                                             ; 3526
                                           6E   DD 00225          PUSHL   STATUS                                                  ;
                                00000000G  00   01 FB 00227          CALLS   #1, LIB$SIGNAL                                          ;
                                           52   D4 0022E 21$:       CLRL    J                                                       ; 3527
                                           23   11 00230          BRB     24$                                                     ;
                                           50   FF A248 9E 00232 22$:  MOVAB   -1(J)[START_LINE], R0                                  ; 3529
                                00 00000000'  EF   50 E5 00237          BBCC    R0, OLD_VALID, 23$                                      ;
                                00000000G  00   00 FB 0023F 23$:       CALLS   #0, SCR$DOWN_SCROLL                                     ; 3530
                                           6E   50 DO 00246          MOVL    R0, STATUS                                              ;
                                           09   6E E8 00249          BLBS    STATUS, 24$                                             ; 3531
                                           6E   DD 0024C          PUSHL   STATUS                                                  ;
                                00000000G  00   01 FB 0024E          CALLS   #1, LIB$SIGNAL                                          ;
                             D8            52         10    AE   F3 00255 24$:  AOBLEQ  SCROLL_AMOUNT, J, 22$                          ; 3527
                                           14    AE   D4 0025A 25$:  CLRL    REGSIZE                                                 ; 3541
                FDF0       57            01         14    F1 0025D 26$:  ACBL    #20, #1, I, 3$                                          ; 3402
                                           0E 00000000'  EF   E8 00263          BLBS    DBG$GL_VT100_FLAG, 29$                          ; 3551
                                           50   D4 0026A          CLRL    I                                                       ; 3554
                             00 00000000'  EF   50 E5 0026C 27$:  BBCC    I, OLD_VALID, 28$                                       ; 3555
                             F4            50   14 F3 00274 28$:  AOBLEQ  #20, I, 27$                                             ;
                                           50   14 D0 00278 29$:  MOVL    #20, I                                                  ; 3565
                                     1C    AE   50 D0 0027B 30$:  MOVL    I, LAST_USED_LINE                                       ; 3567
                             51            50   0C C5 0027F          MULL3   #12, I, R1                                              ; 3568
                                           01 00000000'EF41   91 00283          CMPB    PASTEBOARD[R1], #1                              ;
                                           03   12 0028B          BNEQ    31$                                                     ;
                                           EB   50 F4 0028D          SOBGEQ  I, 30$                                                  ; 3565
                                           56   D4 00290 31$:       CLRL    I                                                       ; 3578
                                        14 AE   D4 00292 32$:       CLRL    RENDITION_LINE                                          ; 3580
                             5A            56   0C C5 00295          MULL3   #12, I, R10                                             ; 3586
                                     0C    AE 00000000'EF4A   9E 00299          MOVAB   PASTEBOARD[R10], 12(SP)                         ;
                             04            01   0C BE 8F 002A2          CASEB   @12(SP), #1, #4                                         ;
                FDB4      00A8                0035      0021     002A7 33$:  .WORD   34$-33$,-                                       ;
                                                      01B9     002AF               35$-33$,-                                       ;
                                                                                   42$-33$,-                                       ;
                                                                                   44$-33$,-                                       ;
                                                                                   52$-33$                                         ;
                                           00000000'  EF   9F 002B1          PUSHAB  P.AHK                                           ; 3734
                                           01   DD 002B7          PUSHL   #1                                                      ;
                                     00028362   8F   DD 002B9          PUSHL   #164706                                                 ;
                                00000000G  00   03 FB 002BF          CALLS   #3, LIB$SIGNAL                                          ;
                                           37   11 002C6          BRB     36$                                                     ;
```

```
                24   AE      FF7C    CD   9E 002C8 34$:    MOVAB   BUFFER, TEXTPTR                    ; 3598
                              20   AE  D4 002CE             CLRL    TEXTCNT                          ; 3599
                1C   AE              56  D1 002D1             CMPL    I, LAST_USED_LINE                ; 3600
                                     76  15 002D5             BLEQ    41$
                              24   AE  D4 002D7             CLRL    TEXTPTR
                                     7C  11 002DA             BRB     43$                              ; 3586
                00000000'EF  4A      9F 002DC 35$:    PUSHAB  PASTEBOARD+8[R10]                    ; 3616
                5B                   9E D0 002E3             MOVL    @(SP)+, DLEPTR
      16        09   AB              01 E1 002E6             BBC     #1, 9(DLEPTR), 37$               ; 3617
                          FF7C    CD   9F 002EB             PUSHAB  BUFFER                           ; 3619
                              24   AE  9F 002EF             PUSHAB  TEXTCNT
                              2C   AE  9F 002F2             PUSHAB  TEXTPTR
                                   5B  DD 002F5             PUSHL   DLEPTR
                              14   AE  DD 002F7             PUSHL   DISPTR
                0000V         CF         05  FB C002FA             CALLS   #5, FORMAT_SOURCE_LINE
                                     57  11 002FF 36$:    BRB     43$
      50        04   AE              1A  C1 00301 37$:    ADDL3   #26, DISPTR, R0                  ; 3624
                10   AE              60  3C 00306             MOVZWL  (R0), DCOL
                24   AE         0C   AB  9E 0030A             MOVAB   12(R11), TEXTPTR                 ; 3625
                12        09    AB  E9 0030F             BLBC    9(DLEPTR), 38$                   ; 3626
                14   AE              01  D0 00313             MOVL    #1, RENDITION_LINE               ; 3629
                     50         24   BE  9A 00317             MOVZBL  @TEXTPTR, R0                     ; 3630
                     50         10   AE  C0 0031B             ADDL2   DCOL, R0
                08   AE         24 BE40  9E 0031F             MOVAB   @TEXTPTR[R0], RENDPTR
                     50         24   BE  9A 00325 38$:    MOVZBL  @TEXTPTR, R0                     ; 3634
                     50         10   AE  C2 00329             SUBL2   DCOL, R0
                                     50  D6 0032D             INCL    R0
                                     02  18 0032F             BGEQ    39$
                                     50  D4 00331             CLRL    R0
                51 00000000G         00  D0 00333 39$:    MOVL    DBG$SRC_TERM_WIDTH, R1
                50                   51  D1 0033A             CMPL    R1, R0
                                     03  15 0033D             BLEQ    40$
                51                   50  D0 0033F             MOVL    R0, R1
          20    AE                   51  D0 00342 40$:    MOVL    R1, TEXTCNT                      ; 3633
                                     10  13 00346             BEQL    43$                              ; 3635
          24    AE         10    AE  C0 00348             ADDL2   DCOL, TEXTPTR
                                     09  11 0034D 41$:    BRB     43$                              ; 3586
          24    AE    FF7C    CD   9E 0034F 42$:    MOVAB   BUFFER, TEXTPTR                  ; 3645
                              20   AE  D4 00355             CLRL    TEXTCNT                          ; 3646
                                011C  31 00358 43$:    BRW     53$                              ; 3586
          24    AE    FF7C    CD   9E 0035B 44$:    MOVAB   BUFFER, TEXTPTR                  ; 3661
          20    AE 00000000G         00  D0 00361             MOVL    DBG$SRC_TERM_WIDTH, TEXTCNT      ; 3662
20  AE              2D                 6E  00  2C 00369             MOVC5   #0, (SP), #45, TEXTCNT, BUFFER   ; 3663
                          FF7C    CD       0036F
                00000000'EF  4A      9F 00372             PUSHAB  PASTEBOARD+4[R10]                ; 3664
          04    AE                   9E  D0 00379             MOVL    @(SP)+, DISPTR
     28   AE    04   AE 0000004C     8F  C1 0037D             ADDL3   #76, DISPTR, NAMEPTR             ; 3665
                     50         28   AE  D0 00387             MOVL    NAMEPTR, R0                      ; 3666
                     5A              60  9A 0038B             MOVZBL  (R0), R10
          50    8F                   5A  91 0038E             CMPB    R10, #80
                                     04  1B 00392             BLEQU   45$
                     5A         50   8F  9A 00394             MOVZBL  #80, R10
          FF7E CD   01   A0              5A  28 00398 45$:    MOVC3   R10, 1(R0), BUFFER+2
                     57         02   AA  9E 0039F             MOVAB   2(R10), N                        ; 3667
                50   04   AE              08  C1 003A3             ADDL3   #8, DISPTR, R0                   ; 3673
                     03                   60  91 003A8             CMPB    (R0), #3
                                     42  12 003AB             BNEQ    47$
```

```
                50        04   AE              34  C1 003AD          ADDL3   #52, DISPTR, R0                          ; 3674
                                                60  D5 003B2          TSTL    (R0)                                     ;
                                                39  13 003B4          BEQL    47$                                      ;
          FF7C CD47 00000000'   EF              09  28 003B6          MOVC3   #9, P.AHF, BUFFER[N]                     ; 3677
                                57              09  C0 003C1          ADDL2   #9, N                                    ; 3678
                                            28  AE  9F 003C4          PUSHAB  NAMEPTR                                  ; 3679
                52        08   AE              34  C1 003C7          ADDL3   #52, DISPTR, R2                          ;
                                                62  DD 003CC          PUSHL   (R2)                                     ;
                   00000000G  00              02  FB 003CE          CALLS   #2, DBG$STA_SYMNAME                      ;
                              50            28  AE  D0 003D5          MOVL    NAMEPTR, R0                              ; 3680
                              5A                60  9A 003D9          MOVZBL  (R0), R10                                ;
                              20                5A  91 003DC          CMPB    R10, #32                                 ;
                                                03  1B 003DF          BLEQU   46$                                      ;
                              5A                20  D0 003E1          MOVL    #32, R10                                 ;
          FF7C CD47           01    A0          5A  28 003E4 46$:     MOVC3   R10, 1(R0), BUFFER[N]                    ;
                              57                5A  C0 003EC          ADDL2   R10, N                                   ; 3681
                              50            02  A7  9E 003EF 47$:     MOVAB   2(R7), R0                                ; 3688
                              32                50  D1 003F3          CMPL    R0, #50                                  ;
                                                03  15 003F6          BLEQ    48$                                      ;
                              50                32  D0 003F8          MOVL    #50, R0                                  ;
                              57                50  D0 003FB 48$:     MOVL    R0, N                                    ;
                   00000000'  EF            04  AE  D1 003FE          CMPL    DISPTR, DBG$SCR_CURDISP_INPUT            ; 3689
                                                0E  12 00406          BNEQ    49$                                      ;
          FF7C CD47 00000000'   EF              07  28 00408          MOVC3   #7, P.AHG, BUFFER[N]                     ; 3692
                                57              06  C0 00413          ADDL2   #6, N                                    ; 3693
                   00000000'  EF            04  AE  D1 00416 49$:     CMPL    DISPTR, DBG$SCR_CURDISP_OUTPUT           ; 3696
                                                0E  12 0041E          BNEQ    50$                                      ;
          FF7C CD47 00000000'   EF              08  28 00420          MOVC3   #8, P.AHH, BUFFER[N]                     ; 3699
                                57              07  C0 0042B          ADDL2   #7, N                                    ; 3700
                   00000000'  EF            04  AE  D1 0042E 50$:     CMPL    DISPTR, DBG$SCR_CURDISP_SOURCE           ; 3703
                                                0E  12 00436          BNEQ    51$                                      ;
          FF7C CD47 00000000'   EF              08  28 00438          MOVC3   #8, P.AHI, BUFFER[N]                     ; 3706
                                57              07  C0 00443          ADDL2   #7, N                                    ; 3707
                   00000000'  EF            04  AE  D1 00446 51$:     CMPL    DISPTR, DBG$SCR_CURDISP_SCROLL           ; 3710
                                                27  12 0044E          BNEQ    53$                                      ;
          FF7C CD47 00000000'   EF              08  28 00450          MOVC3   #8, P.AHJ, BUFFER[N]                     ; 3713
                                57              07  C0 0045B          ADDL2   #7, N                                    ; 3714
                                                17  11 0045E          BRB     53$                                      ; 3586
                         24   AE      FF7C     CD  9E 00460 52$:     MOVAB   BUFFER, TEXTPTR                          ; 3725
                         20   AE 00000000G     00  D0 00466          MOVL    DBG$SRC_TERM_WIDTH, TEXTCNT              ; 3726
    20   AE                2D                6E  00  2C 0046E          MOVC5   #0, (SP), #45, TEXTCNT, BUFFER           ; 3727
                                        FF7C    CD      00474
                                          24    AE  D5 00477 53$:     TSTL    TEXTPTR                                  ; 3743
                                                0B  12 0047A          BNEQ    55$                                      ;
                   00 00000000'  EF                56  E5 0047C          BBCC    I, OLD_VALID, 54$                        ; 3745
                                            009F    31  00484 54$:     BRW     59$                                      ;
                              14        14    AE  E8 00487 55$:     BLBS    RENDITION_LINE, 56$                      ; 3759
                         08   AE      FEF8     CD  9E 0048B          MOVAB   REND_VECTOR, RENDPTR                     ; 3762
       50        0C   BE                08      08  EF 00491          EXTZV   #8, #8, @12(SP), R0                      ; 3763
    20   AE           50                6E      00  2C 00497          MOVC5   #0, (SP), R0, TEXTCNT, @RENDPTR          ;
                                          08    BE      0049D
                                          18    AE  D4 0049F 56$:     CLRL    OLDTEXT                                  ; 3769
                   11 00000000'  EF                56  E1 004A2          BBC     I, OLD_VALID, 57$                        ; 3770
                              5A            56 00000084   8F  C5 004AA          MULL3   #132, I, R10                     ;
                         18   AE 00000000'EF4A            9E  004B2          MOVAB   OLD_SCREEN[R10], OLDTEXT         ;
                                          01    A6  9F 004BB 57$:     PUSHAB  1(I)                                     ; 3772
                              5A            56 00000084   8F  C5 004BE          MULL3   #132, I, R10                     ;
```

```
                                00000000'EF4A  9F  004C6          PUSHAB  OLD_SCREEN_REND[R10]                                    ;
                                      20   AE  DD  004CD          PUSHL   OLDTEXT                                                ;
                                00000000'EF46  DD  004D0          PUSHL   OLD_CNT[I]                                             ;
                                      18   AE  DD  004D7          PUSHL   RENDPTR                                                ;
                                      38   AE  DD  004DA          PUSHL   TEXTPTR                                                ;
                                      38   AE  DD  004DD          PUSHL   TEXTCNT                                                ;
                                    F99A   CF      07  FB  004E0  CALLS   #7, DBG$SCR_OUTPUT_LINE_MINIMAL                        ;
                00000000'EF4A         24   BE      20  AE  28  004E5  MOVC3   TEXTCNT, @TEXTPTR, OLD_SCREEN[R10]              ; 3777
                00000000'EF4A         08   BE      20  AE  28  004F0  MOVC3   TEXTCNT, @RENDPTR, OLD_SCREEN_REND[R10]        ; 3778
                          00000000'EF46      20  AE  D0  004FB  MOVL    TEXTCNT, OLD_CNT[I]                                    ; 3779
          50        0C   BE                 08      08  EF  00504  EXTZV   #8, #8, @12(SP), R0                                ; 3780
                          00000000'EF46          50  90  0050A  MOVB    R0, OLD_REND[I]
                                          08  14  AE  E9  00512  BLBC    RENDITION_LINE, 58$                                    ; 3781
                          00000000'EF46          01  8E  00516  MNEGB   #1, OLD_REND[I]
                                00  00000000'   EF      56  E2  0051E  58$:  BBSS    I, OLD_VALID, 59$                         ; 3782
              FD66              56              01      14  F1  00526  59$:  ACBL    #20, #-, I, 32$                           ; 3578
                                                58      16  D0  0052C  MOVL    #22, START_LINE                                   ; 3792
                                                59      18  D0  0052F  MOVL    #24, END_LINE                                     ; 3793
                                                50      15  D0  00532  MOVL    #21, I                                            ; 3794
                                              51        50      0C  C5  00535  60$:  MULL3   #12, I, R1                        ; 3796
                                                01  00000000'EF41  91  00539  CMPB    PASTEBOARD-12[R1], #1
                                                        06      12  00541  BNEQ    61$
                                                58      50      D0  00543  MOVL    I, START_LINE                                 ; 3800
                                                EC      50      F5  00546  SOBGTR   I, 60$                                       ; 3794
                                                7E      58      7D  00549  61$:  MOVQ    START_LINE, -(SP)                     ; 3803
                                  00000000G     00      02      FB  0054C  CALLS   #2, SCR$SET_SCROLL
                                                6E      50      D0  00553  MOVL    R0, STATUS
                                                09      6E      E8  00556  BLBS    STATUS, 62$                                   ; 3804
                                                6E      6E      DD  00559  PUSHL   STATUS
                                  00000000G     00      01      FB  0055B  CALLS   #1, LIB$SIGNAL
                                                        01      DD  00562  62$:  PUSHL   #1                                    ; 3809
                                                        18      DD  00564  PUSHL   #24
                                  00000000G     00      02      FB  00566  CALLS   #2, SCR$SET_CURSOR
                                                6E      50      D0  0056D  MOVL    R0, STATUS
                                                09      6E      E8  00570  BLBS    STATUS, 63$                                   ; 3810
                                                6E      6E      DD  00573  PUSHL   STATUS
                                  00000000G     00      01      FB  00575  CALLS   #1, LIB$SIGNAL
                                                        7E      D4  0057C  63$:  CLRL    -(SP)                                 ; 3816
                                  00000000G     00      01      FB  0057E  CALLS   #1, SCR$PUT_BUFFER
                                                6E      50      D0  00585  MOVL    R0, STATUS
                                                09      6E      E8  00588  BLBS    STATUS, 64$                                   ; 3817
                                                6E      6E      DD  0058B  PUSHL   STATUS
                                  00000000G     00      01      FB  0058D  CALLS   #1, LIB$SIGNAL
                                                05  00000000'   EF  E9  00594  64$:  BLBC    DBG$GL_SCREEN_LOG, 65$                ; 3823
                                    0000V       CF      00      FB  0059B  CALLS   #0, DBG$SCR_SCREEN_TO_LOGFILE
                                                        04      005A0  65$:  RET                                                  ; 3826

; Routine Size:  1441 bytes,    Routine Base:  DBG$CODE + 1D33
```

```
;  3722          3827  1 GLOBAL ROUTINE DBG$SCR_PARSE_CANDISP_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  3723          3828  1 !
;  3724          3829  1 ! FUNCTION
;  3725          3830  1 !       This routine parses the CANCEL DISPLAY command. It accepts a command
;  3726          3831  1 !       line string descriptor as input and produces a Verb Node for the parsed
;  3727          3832  1 !       CANCEL DISPLAY command as output.  That Verb Node and its attached Noun
;  3728          3833  1 !       Nodes later serve as input to DBG$SCR_EXECUTE_CANDISP_CMD which actual-
;  3729          3834  1 !       ly executes the command.
;  3730          3835  1 !
;  3731          3836  1 ! INPUTS
;  3732          3837  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  3733          3838  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  3734          3839  1 !                    first character after the CANCEL DISPLAY keywords.
;  3735          3840  1 !
;  3736          3841  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  3737          3842  1 !                    being parsed.
;  3738          3843  1 !
;  3739          3844  1 ! OUTPUTS
;  3740          3845  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  3741          3846  1 !                    first character after the end of the command.  This normally
;  3742          3847  1 !                    means that the input string is exhausted.
;  3743          3848  1 !
;  3744          3849  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it and all
;  3745          3850  1 !                    its attached Noun Nodes contains all information picked up
;  3746          3851  1 !                    during the parse of the CANCEL DISPLAY command.
;  3747          3852  1 !
;  3748          3853  1
;  3749          3854  2     BEGIN
;  3750          3855  2
;  3751          3856  2     MAP
;  3752          3857  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  3753          3858  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  3754          3859  2
;  3755          3860  2     LOCAL
;  3756          3861  2         DISPID: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;  3757          3862  2         NAMEPTR,                        ! Pointer to ASCIC display name
;  3758          3863  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Pointer to current Noun Node
;  3759          3864  2         OLDPTR: REF DBG$NOUN_NODE;      ! Pointer to previous Noun Node in list
;  3760          3865  2
;  3761          3866  2
;  3762          3867  2
;  3763          3868  2     ! First check for the /ALL qualifier.  If it is present, we return right
;  3764          3869  2     ! away with zero Noun Nodes attached to the Verb Node.  No additional
;  3765          3870  2     ! parameters are allowed in this case.
;  3766          3871  2     !
;  3767          3872  2     VERB_NODE[DBG$L_VERB_OBJECT_PTR] = 0;
;  3768          3873  2     IF DBG$NMATCH(.INPUT_DESC, DBG$CS_SLASH, 1)
;  3769          3874  2     THEN
;  3770          3875  3         BEGIN
;  3771          3876  3         IF DBG$NMATCH(.INPUT_DESC, DBG$CS_ALL, 1) THEN RETURN;
;  3772          3877  3         DBG$SYNTAX_ERROR(.INPUT_DESC);
;  3773          3878  2         END;
;  3774          3879  2
;  3775          3880  2
;  3776          3881  2     ! Otherwise, go into a loop to pick up the names of all displays to be
;  3777          3882  2     ! cancelled.  For each such display name, look up the corresponding
;  3778          3883  2     ! Display ID (thus ensuring that the display name is valid) and construct
```

```
;  3779          3884  2     ! a Noun Node to hold a pointer to the display name.
;  3780          3885  2     !
;  3781          3886  2     NOUN_NODE = 0;
;  3782          3887  2     WHILE TRUE DO
;  3783          3888  2         BEGIN
;  3784          3889  3
;  3785          3890  3
;  3786          3891  3         ! Pick up the display name and look it up in the Display List to make
;  3787          3892  3         ! sure there is such a display.
;  3788          3893  3         !
;  3789          3894  3         NAMEPTR = PARSE_DISPLAY_NAME(.INPUT_DESC, FALSE);
;  3790          3895  3         DISPID = DBG$SCR_LOOKUP_DISPLAY(.NAMEPTR);
;  3791          3896  3         IF .DISPID EQL 0 THEN SIGNAL(DBG$_NOSUCHDISP, 1, .NAMEPTR);
;  3792          3897  3
;  3793          3898  3
;  3794          3899  3         ! The display name is good.  Create a new Noun Node to hold a pointer
;  3795          3900  3         ! to the display name and add that node to the Noun Node list attached
;  3796          3901  3         ! to the Verb Node.
;  3797          3902  3         !
;  3798          3903  3         OLDPTR = .NOUN_NODE;
;  3799          3904  3         NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE);
;  3800          3905  3         NOUN_NODE[DBG$L_NOUN_VALUE] = .NAMEPTR;
;  3801          3906  3         IF .OLDPTR EQL 0
;  3802          3907  3         THEN
;  3803          3908  3             VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE
;  3804          3909  3
;  3805          3910  3         ELSE
;  3806          3911  3             OLDPTR[DBG$L_NOUN_LINK] = .NOUN_NODE;
;  3807          3912  3
;  3808          3913  3
;  3809          3914  3         ! Check that the display name is properly followed by a comma (in which
;  3810          3915  3         ! case we loop for more) or a carriage return (which ends the command).
;  3811          3916  3         !
;  3812          3917  3         IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1) THEN EXITLOOP;
;  3813          3918  3         IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COMMA, 1)
;  3814          3919  3         THEN
;  3815          3920  3             DBG$SYNTAX_ERROR(.INPUT_DESC);
;  3816          3921  3
;  3817          3922  2         END;                            ! End of loop over display names
;  3818          3923  2
;  3819          3924  2
;  3820          3925  2     ! The command is fully parsed and the Verb Node, complete with its list
;  3821          3926  2     ! of Noun Nodes, is fully constructed.  Now return.
;  3822          3927  2     !
;  3823          3928  2     RETURN;
;  3824          3929  2
;  3825          3930  1     END;


                                07FC 00000          .ENTRY  DBG$SCR_PARSE_CANDISP_CMD, Save R2,R3,R4,-  ; 3827
                                                            R5,R6,R7,R8,R9,R10                              ;
                    5A 00000000G   00   9E 00002    MOVAB   DBG$SYNTAX_ERROR, R10                           ;
                    59 00000000'   EF   9E 00009    MOVAB   DBG$CS_SLASH, R9                                ;
                    58 00000000G   00   9E 00010    MOVAB   DBG$NMATCH, R8                                  ;
```

```
                52        08    AC  D0 00017          MOVL    VERB_NODE, R2                   ; 3872
                          08    A2  D4 0001B          CLRL    8(R2)
                          01    DD  0001E             PUSHL   #1                              ; 3873
                          59    DD  00020             PUSHL   R9
                53        04    AC  D0 00022          MOVL    INPUT_DESC, R3
                          53    DD  00026             PUSHL   R3
                68              03  FB 00028         CALLS   #3, DBG$NMATCH
                13              50  E9 0002B         BLBC    R0, 1$
                          01    DD  0002E             PUSHL   #1                              ; 3876
                      FF5E      C9  9F 00030          PUSHAB  DBG$CS_ALL
                          53    DD  00034             PUSHL   R3
                68              03  FB 00036         CALLS   #3, DBG$NMATCH
                73              50  E8 00039         BLBS    R0, 6$
                          53    DD  0003C             PUSHL   R3                              ; 3877
                6A              01  FB 0003E         CALLS   #1, DBG$SYNTAX_ERROR
                          55    D4  00041 1$:         CLRL    NOUN_NODE                       ; 3886
                          7E    D4  00043 2$:         CLRL    -(SP)                           ; 3894
                          53    DD  00045             PUSHL   R3
          0000V CF              02  FB 00047         CALLS   #2, PARSE_DISPLAY_NAME
                56              50  D0 0004C         MOVL    R0, NAMEPTR
                          56    DD  0004F             PUSHL   NAMEPTR                         ; 3895
          F702  CF              01  FB 00051         CALLS   #1, DBG$SCR_LOOKUP_DISPLAY
                57              50  D0 00056         MOVL    R0, DISPID
                                11  12 00059         BNEQ    3$                              ; 3896
                          56    DD  0005B             PUSHL   NAMEPTR
                          01    DD  0005D             PUSHL   #1
                00028A62        8F  DD 0005F         PUSHL   #166498
    00000000G   00              03  FB 00065         CALLS   #3, LIB$SIGNAL
                54              55  D0 0006C 3$:      MOVL    NOUN_NODE, OLDPTR               ; 3903
                                04  DD 0006F         PUSHL   #4                              ; 3904
    00000000G   00              01  FB 00071         CALLS   #1, DBG$GET_TEMPMEM
                55              50  D0 00078         MOVL    R0, NOUN_NODE
                65              56  D0 0007B         MOVL    NAMEPTR, (NOUN_NODE)            ; 3905
                                54  D5 0007E         TSTL    OLDPTR                          ; 3906
                                06  12 00080         BNEQ    4$
                      08    A2        55  D0 00082   MOVL    NOUN_NODE, 8(R2)                ; 3908
                                04  11 00086         BRB     5$
                      08    A4        55  D0 00088 4$:      MOVL    NOUN_NODE, 8(OLDPTR)    ; 3911
                          01    DD  0008C 5$:         PUSHL   #1                              ; 3917
                      FF79      C9  9F 0008E          PUSHAB  DBG$CS_CR
                          53    DD  00092             PUSHL   R3
                68              03  FB 00094         CALLS   #3, DBG$NMATCH
                15              50  E8 00097         BLBS    R0, 6$
                          01    DD  0009A             PUSHL   #1                              ; 3918
                      FF77      C9  9F 0009C          PUSHAB  DBG$CS_COMMA
                          53    DD  000A0             PUSHL   R3
                68              03  FB 000A2         CALLS   #3, DBG$NMATCH
                9B              50  E8 000A5         BLBS    R0, 2$
                          53    DD  000A8             PUSHL   R3                              ; 3920
                6A              01  FB 000AA         CALLS   #1, DBG$SYNTAX_ERROR
                                94  11 000AD         BRB     2$                              ; 3887
                                    04 000AF 6$:      RET                                     ; 3930

; Routine Size:  176 bytes,    Routine Base:  DBG$CODE + 22D4
```

```
;  3827          3931  1 GLOBAL ROUTINE DBG$SCR_PARSE_CANWIND_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  3828          3932  1 !
;  3829          3933  1 ! FUNCTION
;  3830          3934  1 !       This routine parses the CANCEL WINDOW command. It accepts a command
;  3831          3935  1 !       line string descriptor as input and produces a Verb Node for the parsed
;  3832          3936  1 !       CANCEL WINDOW command as output.  That Verb Node and its attached Noun
;  3833          3937  1 !       Nodes later serve as input to DBG$SCR_EXECUTE_CANWIND_CMD which actual-
;  3834          3938  1 !       ly executes the command.
;  3835          3939  1 !
;  3836          3940  1 ! INPUTS
;  3837          3941  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  3838          3942  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  3839          3943  1 !                    first character after the CANCEL WINDOW keywords.
;  3840          3944  1 !
;  3841          3945  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  3842          3946  1 !                   being parsed.
;  3843          3947  1 !
;  3844          3948  1 ! OUTPUTS
;  3845          3949  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  3846          3950  1 !                    first character after the end of the command.  This normally
;  3847          3951  1 !                    means that the input string is exhausted.
;  3848          3952  1 !
;  3849          3953  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it and all
;  3850          3954  1 !                   its attached Noun Nodes contains all information picked up
;  3851          3955  1 !                   during the parse of the CANCEL WINDOW command.
;  3852          3956  1 !
;  3853          3957  1
;  3854          3958  2     BEGIN
;  3855          3959  2
;  3856          3960  2     MAP
;  3857          3961  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  3858          3962  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  3859          3963  2
;  3860          3964  2     LOCAL
;  3861          3965  2         NAMEPTR,                        ! Pointer to ASCIC window name
;  3862          3966  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Pointer to current Noun Node
;  3863          3967  2         OLDPTR: REF DBG$NOUN_NODE,      ! Pointer to previous Noun Node in list
;  3864          3968  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to Screen Window Entry
;  3865          3969  2
;  3866          3970  2
;  3867          3971  2
;  3868          3972  2     ! First check for the /ALL qualifier.  If it is present, we return right
;  3869          3973  2     ! away with zero Noun Nodes attached to the Verb Node.  No additional
;  3870          3974  2     ! parameters are allowed in this case.
;  3871          3975  2     !
;  3872          3976  2     VERB_NODE[DBG$L_VERB_OBJECT_PTR] = 0;
;  3873          3977  2     IF DBG$NMATCH(.INPUT_DESC, DBG$CS_SLASH, 1)
;  3874          3978  2     THEN
;  3875          3979  3         BEGIN
;  3876          3980  3         IF DBG$NMATCH(.INPUT_DESC, DBG$CS_ALL, 1) THEN RETURN;
;  3877          3981  3         DBG$SYNTAX_ERROR(.INPUT_DESC);
;  3878          3982  2         END;
;  3879          3983  2
;  3880          3984  2
;  3881          3985  2     ! Otherwise, go into a loop to pick up the names of all windows to be
;  3882          3986  2     ! cancelled.  For each such window name, look up the corresponding Screen
;  3883          3987  2     ! Window Entry (thus ensuring that the window name is valid) and construct
```

```
;  3884          3988  2      ! a Noun Node to hold a pointer to the window name.
;  3885          3989  2      !
;  3886          3990  2      NOUN_NODE = 0;
;  3887          3991  2      WHILE TRUE DO
;  3888          3992  3          BEGIN
;  3889          3993  3
;  3890          3994  3
;  3891          3995  3          ! Pick up the window name and look it up in the Screen Window List to
;  3892          3996  3          ! make sure there exists such a window.
;  3893          3997  3          !
;  3894          3998  3          NAMEPTR = PARSE_WINDOW_NAME(.INPUT_DESC);
;  3895          3999  3          WPTR = DBG$SCR_LOOKUP_WINDOW(.NAMEPTR);
;  3896          4000  3          IF .WPTR EQL 0 THEN SIGNAL(DBG$_NOSUCHWIND, 1, .NAMEPTR);
;  3897          4001  3
;  3898          4002  3
;  3899          4003  3          ! The window name is good.  Create a new Noun Node to hold a pointer to
;  3900          4004  3          ! the window name and add that node to the Noun Node list attached to
;  3901          4005  3          ! the Verb Node.
;  3902          4006  3          !
;  3903          4007  3          OLDPTR = .NOUN_NODE;
;  3904          4008  3          NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE);
;  3905          4009  3          NOUN_NODE[DBG$L_NOUN_VALUE] = .NAMEPTR;
;  3906          4010  3          IF .OLDPTR EQL 0
;  3907          4011  3          THEN
;  3908          4012  3              VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE
;  3909          4013  3
;  3910          4014  3          ELSE
;  3911          4015  3              OLDPTR[DBG$L_NOUN_LINK] = .NOUN_NODE;
;  3912          4016  3
;  3913          4017  3
;  3914          4018  3          ! Check that the window name is properly followed by a comma (in which
;  3915          4019  3          ! case we loop for more) or a carriage return (which ends the command).
;  3916          4020  3          !
;  3917          4021  3          IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1) THEN EXITLOOP;
;  3918          4022  3          IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COMMA, 1)
;  3919          4023  3          THEN
;  3920          4024  3              DBG$SYNTAX_ERROR(.INPUT_DESC);
;  3921          4025  3
;  3922          4026  2          END;                            ! End of loop over window names
;  3923          4027  2
;  3924          4028  2
;  3925          4029  2      ! The command is fully parsed and the Verb Node, complete with its list
;  3926          4030  2      ! of Noun Nodes, is fully constructed.  Now return.
;  3927          4031  2      !
;  3928          4032  2      RETURN;
;  3929          4033  2
;  3930          4034  1      END;


                                07FC 00000          .ENTRY  DBG$SCR_PARSE_CANWIND_CMD, Save R2,R3,R4,-  ; 3931
                                                            R5,R6,R7,R8,R9,R10                              ;
                    5A 00000000G   00   9E 00002    MOVAB   DBG$SYNTAX_ERROR, R10                           ;
                    59 00000000'   EF   9E 00009    MOVAB   DBG$CS_SLASH, R9                                ;
                    58 00000000G   00   9E 00010    MOVAB   DBG$NMATCH, R8                                  ;
```

```
                52      08  AC  D0 00017          MOVL    VERB_NODE, R2                   ; 3976
                        08  A2  D4 0001B          CLRL    8(R2)
                            01  DD 0001E          PUSHL   #1                              ; 3977
                            59  DD 00020          PUSHL   R9
                53      04  AC  D0 00022          MOVL    INPUT_DESC, R3
                            53  DD 00026          PUSHL   R3
                68              03  FB 00028      CALLS   #3, DBG$NMATCH
                13              50  E9 0002B      BLBC    R0, 1$
                                01  DD 0002E      PUSHL   #1                              ; 3980
                        FF5E    C9  9F 00030      PUSHAB  DBG$CS_ALL
                                53  DD 00034      PUSHL   R3
                68              03  FB 00036      CALLS   #3, DBG$NMATCH
                71              50  E8 00039      BLBS    R0, 6$
                                53  DD 0003C      PUSHL   R3                              ; 3981
                6A              01  FB 0003E      CALLS   #1, DBG$SYNTAX_ERROR
                                55  D4 00041  1$: CLRL    NOUN_NODE                       ; 3990
                                53  DD 00043  2$: PUSHL   R3                              ; 3998
          0000V CF              01  FB 00045      CALLS   #1, PARSE_WINDOW_NAME
                56              50  D0 0004A      MOVL    R0, NAMEPTR
                                56  DD 0004D      PUSHL   NAMEPTR                         ; 3999
          F79E  CF              01  FB 0004F      CALLS   #1, DBG$SCR_LOOKUP_WINDOW
                57              50  D0 00054      MOVL    R0, WPTR
                                11  12 00057      BNEQ    3$                              ; 4000
                                56  DD 00059      PUSHL   NAMEPTR
                                01  DD 0005B      PUSHL   #1
                    00028A6A    8F  DD 0005D      PUSHL   #166506
 00000000G      00              03  FB 00063      CALLS   #3, LIB$SIGNAL
                54              55  D0 0006A  3$: MOVL    NOUN_NODE, OLDPTR               ; 4007
                                04  DD 0006D      PUSHL   #4                              ; 4008
 00000000G      00              01  FB 0006F      CALLS   #1, DBG$GET_TEMPMEM
                55              50  D0 00076      MOVL    R0, NOUN_NODE
                65              56  D0 00079      MOVL    NAMEPTR, (NOUN_NODE)            ; 4009
                                54  D5 0007C      TSTL    OLDPTR                          ; 4010
                                06  12 0007E      BNEQ    4$
                    08  A2          55  D0 00080  MOVL    NOUN_NODE, 8(R2)                ; 4012
                                04  11 00084      BRB     5$
                    08  A4          55  D0 00086  4$: MOVL    NOUN_NODE, 8(OLDPTR)        ; 4015
                                01  DD 0008A  5$: PUSHL   #1                              ; 4021
                        FF79    C9  9F 0008C      PUSHAB  DBG$CS_CR
                                53  DD 00090      PUSHL   R3
                68              03  FB 00092      CALLS   #3, DBG$NMATCH
                15              50  E8 00095      BLBS    R0, 6$
                                01  DD 00098      PUSHL   #1                              ; 4022
                        FF77    C9  9F 0009A      PUSHAB  DBG$CS_COMMA
                                53  DD 0009E      PUSHL   R3
                68              03  FB 000A0      CALLS   #3, DBG$NMATCH
                9D              50  E8 000A3      BLBS    R0, 2$
                                53  DD 000A6      PUSHL   R3                              ; 4024
                6A              01  FB 000A8      CALLS   #1, DBG$SYNTAX_ERROR
                                96  11 000AB      BRB     2$                              ; 3991
                                    04 000AD  6$: RET                                     ; 4034

; Routine Size:  174 bytes,    Routine Base:  DBG$CODE + 2384
```

```
;  3932          4035  1 GLOBAL ROUTINE DBG$SCR_PARSE_DISPLAY_CMD(INPUT_DESC,
;  3933          4036  1                                          SET_FLAG, VERB_NODE): NOVALUE =
;  3934          4037  1 !
;  3935          4038  1 ! FUNCTION
;  3936          4039  1 !       This routine parses the DISPLAY and SET DISPLAY commands.  It accepts
;  3937          4040  1 !       a command line string descriptor as input and produces a verb node
;  3938          4041  1 !       for the parsed command as output.  Since the DISPLAY and SET DISPLAY
;  3939          4042  1 !       commands are very similar in syntax, both commands are parsed by the
;  3940          4043  1 !       same code.  The SET_FLAG parameter, which is set to TRUE if this is
;  3941          4044  1 !       a SET DISPLAY command, is used to special-case those features which
;  3942          4045  1 !       differ between the two commands.
;  3943          4046  1 !
;  3944          4047  1 ! INPUTS
;  3945          4048  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  3946          4049  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  3947          4050  1 !                    first character after the DISPLAY or SET DISPLAY keyword.
;  3948          4051  1 !
;  3949          4052  1 !       SET_FLAG - This flag has the value TRUE if the command being parsed is
;  3950          4053  1 !                  a SET DISPLAY command and the value FALSE if the command
;  3951          4054  1 !                  being parsed is a DISPLAY command.
;  3952          4055  1 !
;  3953          4056  1 !       VERB_NODE - A pointer to the verb node to be built up for the command
;  3954          4057  1 !                   being parsed.  The verb node is assumed to have one attached
;  3955          4058  1 !                   noun node but no adverb node on input.
;  3956          4059  1 !
;  3957          4060  1 ! OUTPUTS
;  3958          4061  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  3959          4062  1 !                    first character after the end of the command.  This normally
;  3960          4063  1 !                    means that the input string is exhausted.
;  3961          4064  1 !
;  3962          4065  1 !       VERB_NODE - The passed-in verb node is filled in so that it and all
;  3963          4066  1 !                   its attached adverb and noun nodes contains all information
;  3964          4067  1 !                   picked up during the parse of the command.
;  3965          4068  1 !
;  3966          4069  1
;  3967          4070  2     BEGIN
;  3968          4071  2
;  3969          4072  2     MAP
;  3970          4073  2         INPUT_DESC: REF BLOCK[,BYTE],    ! Pointer to input string descriptor
;  3971          4074  2         VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to input verb node
;  3972          4075  2
;  3973          4076  2     OWN
;  3974          4077  2         DW_INDEX: INITIAL(0);            ! Default window index (0=H1, 1=H2)
;  3975          4078  2
;  3976          4079  2     LOCAL
;  3977          4080  2         ADVERB_NODE:                     ! Pointer to Adverb Node to contain
;  3978          4081  2                 REF DBG$ADVERB_NODE,     !      qualifier information
;  3979          4082  2         CBEG,                            ! CBEG parameter of window specification
;  3980          4083  2         CLEAR_FLAG,                      ! Flag set if /CLEAR qualifier found
;  3981          4084  2         CLEN,                            ! CLEN parameter of window specification
;  3982          4085  2         CONTENT_PTR,                     ! Pointer to DEBUG command list which
;  3983          4086  2                                          !      defines display contents
;  3984          4087  2         EXPC_DESC: BLOCK[8,BYTE],        ! String descriptor for default source
;  3985          4088  2                                          !      DEBUG command list
;  3986          4089  2         FLAGWORD,                        ! Flag longword with all qualifier
;  3987          4090  2                                          !      information--saved in Adverb
;  3988          4091  2                                          !      Node value field
```

```
;3989     4092  2          GENERATE_ALL_FLAG,              ! Flag set for /GENERATE all displays
;3990     4093  2          GENERATE_FLAG,                  ! Flag set if /GENERATE qualifier found
;3991     4094  2          HIDE_FLAG,                      ! Flag set if /HIDE qualifier found
;3992     4095  2          KIND,                           ! The display kind (content spec kind)
;3993     4096  2          MARK_FLAG,                      ! Flag set if /MARK_CHANGE qualifier found
;3994     4097  2          NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to current ASCIC display name
;3995     4098  2          NOMARK_FLAG,                    ! Flag set if /NOMARK_CHANGE qual found
;3996     4099  2          NOUN_NODE: REF DBG$NOUN_NODE,   ! Pointer to current Noun Node
;3997     4100  2          OLD_NOUN_PTR: REF DBG$NOUN_NODE,! Pointer to previous Noun Node
;3998     4101  2          RBEG,                           ! RBEG parameter of window specification
;3999     4102  2          REFRESH_FLAG,                   ! Flag set if /REFRESH qualifier found
;4000     4103  2          REMOVED_FLAG,                   ! Flag set if /REMOVED qualifier found
;4001     4104  2          RLEN,                           ! RLEN parameter of window specification
;4002     4105  2          SIZE,                           ! Display size from /SIZE:n qualifier
;4003     4106  2          WNAMEPTR: REF VECTOR[,BYTE],    ! Pointer to current ASCIC window name
;4004     4107  2          WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to current Screen Window Entry
;4005     4108  2
;4006     4109  2
;4007     4110  2
;4008     4111  2      ! Initialize various flags used to mark the various qualifiers specified.
;4009     4112  2      !
;4010     4113  2      REFRESH_FLAG = FALSE;
;4011     4114  2      HIDE_FLAG = FALSE;
;4012     4115  2      MARK_FLAG = FALSE;
;4013     4116  2      NOMARK_FLAG = FALSE;
;4014     4117  2      REMOVED_FLAG = FALSE;
;4015     4118  2      CLEAR_FLAG = FALSE;
;4016     4119  2      GENERATE_ALL_FLAG = FALSE;
;4017     4120  2      GENERATE_FLAG = FALSE;
;4018     4121  2      SIZE = 0;
;4019     4122  2
;4020     4123  2
;4021     4124  2      ! Loop to pick up all command qualifiers.  For each qualifier, set the
;4022     4125  2      ! appropriate flag to indicate that it was found.
;4023     4126  2      !
;4024     4127  2      WHILE DBG$NMATCH(.INPUT_DESC, DBG$CS_SLASH, 1) DO
;4025     4128  3          BEGIN                                                              ;
;4026     4129  3
;4027     4130  3
;4028     4131  3          ! Determine which qualifier we have and mark its presence.
;4029     4132  3          !
;4030     4133  3          SELECTONE TRUE OF
;4031     4134  3              SET
;4032     4135  3
;4033     4136  3
;4034     4137  3              ! If this is the /REFRESH qualifier, mark its presence.  Note that
;4035     4138  3              ! this qualifier is allowed only on the DISPLAY command.
;4036     4139  3              !
;4037     4140  3              [IF .SET_FLAG THEN FALSE ELSE
;4038     4141  3               DBG$NMATCH(.INPUT_DESC, DBG$CS_REFRESH, 3)]:
;4039     4142  3                  REFRESH_FLAG = TRUE;
;4040     4143  3
;4041     4144  3
;4042     4145  3              ! If this is the /CLEAR qualifier, mark its presence.  This quali-
;4043     4146  3              ! fier is only allowed on the DISPLAY command.
;4044     4147  3              !
;4045     4148  3              [IF .SET_FLAG THEN FALSE ELSE
```

```
; 4046     4149  3          DBG$NMATCH(.INPUT_DESC, DBG$CS_CLEAR, 2)]:
; 4047     4150  3              CLEAR_FLAG = TRUE;
; 4048     4151  3
; 4049     4152  3
; 4050     4153  3          ! If this is the /GENERATE qualifier, mark its presence.  Note that
; 4051     4154  3          ! this qualifier is allowed only on the DISPLAY command.
; 4052     4155  3          !
; 4053     4156  3          [IF .SET_FLAG THEN FALSE ELSE
; 4054     4157  3           DBG$NMATCH(.INPUT_DESC, DBG$CS_GENERATE, 2)]:
; 4055     4158  3              GENERATE_FLAG = TRUE;
; 4056     4159  3
; 4057     4160  3
; 4058     4161  3          ! If this is the /HIDE qualifier, mark its presence.
; 4059     4162  3          !
; 4060     4163  3          [DBG$NMATCH(.INPUT_DESC, DBG$CS_HIDE, 2)]:
; 4061     4164  3              HIDE_FLAG = TRUE;
; 4062     4165  3
; 4063     4166  3
; 4064     4167  3          ! If this is the /MARK_CHANGE qualifier, mark its presence.
; 4065     4168  3          !
; 4066     4169  3          [DBG$NMATCH(.INPUT_DESC, DBG$CS_MARK_CHANGE, 2)]:
; 4067     4170  4              BEGIN
; 4068     4171  4              MARK_FLAG = TRUE;
; 4069     4172  4              NOMARK_FLAG = FALSE;
; 4070     4173  3              END;
; 4071     4174  3
; 4072     4175  3
; 4073     4176  3          ! If this is the /NOMARK_CHANGE qualifier, mark its presence.  This
; 4074     4177  3          ! qualifier is only allowed on the DISPLAY command.
; 4075     4178  3          !
; 4076     4179  3          [IF .SET_FLAG THEN FALSE ELSE
; 4077     4180  3           DBG$NMATCH(.INPUT_DESC, DBG$CS_NOMARK_CHANGE, 4)]:
; 4078     4181  4              BEGIN
; 4079     4182  4              NOMARK_FLAG = TRUE;
; 4080     4183  4              MARK_FLAG = FALSE;
; 4081     4184  3              END;
; 4082     4185  3
; 4083     4186  3
; 4084     4187  3          ! If this is the /REMOVED qualifier, mark its presence.
; 4085     4188  3          !
; 4086     4189  3          [DBG$NMATCH(.INPUT_DESC, DBG$CS_REMOVED, 3)]:
; 4087     4190  3              REMOVED_FLAG = TRUE;
; 4088     4191  3
; 4089     4192  3
; 4090     4193  3          ! If this is the /SIZE:n qualifier, pick up the requested display
; 4091     4194  3          ! size and check it for validity.
; 4092     4195  3          !
; 4093     4196  3          [DBG$NMATCH(.INPUT_DESC, DBG$CS_SIZE, 2)]:
; 4094     4197  4              BEGIN
; 4095     4198  4              IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COLON, 1)
; 4096     4199  4              THEN
; 4097     4200  5                  BEGIN
; 4098     4201  5                  IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_EQUAL, 1)
; 4099     4202  5                  THEN
; 4100     4203  5                      DBG$SYNTAX_ERROR(.INPUT_DESC);
; 4101     4204  5
; 4102     4205  4                  END;
```

```
;  4103          4206  4
;  4104          4207  4                DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, SIZE);
;  4105          4208  5                IF (.SIZE LSS 1) OR (.SIZE GTR 1000)
;  4106          4209  4                THEN
;  4107          4210  4                    SIGNAL(DBG$_INVDSPSIZ, 1, .SIZE);
;  4108          4211  4
;  4109          4212  3                END;
;  4110          4213  3
;  4111          4214  3
;  4112          4215  3            ! Any other qualifier constitutes a syntax error, which we
;  4113          4216  3            ! signal at this point.
;  4114          4217  3            !
;  4115          4218  3            [OTHERWISE]:
;  4116          4219  3                DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4117          4220  3
;  4118          4221  3            TES;
;  4119          4222  3
;  4120          4223  2        END;                                ! End of qualifier scanning loop
;  4121          4224  2
;  4122          4225  2
;  4123          4226  2        ! For the DISPLAY/GENERATE command, the parameter list is optional.  If
;  4124          4227  2        ! there are no parameters, we set GENERATE_ALL_FLAG to indicate that all
;  4125          4228  2        ! automatically generated displays should be regenerated.
;  4126          4229  2        !
;  4127          4230  2        IF .GENERATE_FLAG AND DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1)
;  4128          4231  2        THEN
;  4129          4232  2            GENERATE_ALL_FLAG = TRUE;
;  4130          4233  2
;  4131          4234  2
;  4132          4235  2        ! Build an Adverb Node and fill all qualifier information into the value
;  4133          4236  2        ! field of that node.
;  4134          4237  2        !
;  4135          4238  2        FLAGWORD = 0;
;  4136          4239  2        FLAGWORD<V_(0)> = .REFRESH_FLAG;
;  4137          4240  2        FLAGWORD<V_(1)> = .CLEAR_FLAG;
;  4138          4241  2        FLAGWORD<V_(2)> = .GENERATE_ALL_FLAG;
;  4139          4242  2        FLAGWORD<V_(3)> = .GENERATE_FLAG;
;  4140          4243  2        FLAGWORD<V_(4)> = .HIDE_FLAG;
;  4141          4244  2        FLAGWORD<V_(5)> = .MARK_FLAG;
;  4142          4245  2        FLAGWORD<V_(6)> = .NOMARK_FLAG;
;  4143          4246  2        FLAGWORD<V_(7)> = .REMOVED_FLAG;
;  4144          4247  2        FLAGWORD<V_(16,16)> = .SIZE;
;  4145          4248  2        ADVERB_NODE = DBG$GET_TEMPMEM(DBG$K_ADVERB_NODE_SIZE);
;  4146          4249  2        ADVERB_NODE[DBG$L_ADVERB_VALUE] = .FLAGWORD;
;  4147          4250  2        VERB_NODE[DBG$L_VERB_ADVERB_PTR] = .ADVERB_NODE;
;  4148          4251  2
;  4149          4252  2
;  4150          4253  2        ! For the DISPLAY/REFRESH command, no further parameters are allowed.  We
;  4151          4254  2        ! thus return right away.  We also return immediately for the parameter-
;  4152          4255  2        ! less DISPLAY/GENERATE command.
;  4153          4256  2        !
;  4154          4257  2        IF .REFRESH_FLAG OR .GENERATE_ALL_FLAG THEN RETURN;
;  4155          4258  2
;  4156          4259  2
;  4157          4260  2        ! Loop to pick up the names and attributes of all displays to be displayed
;  4158          4261  2        ! or created.  For the SET DISPLAY command, we only go through this loop
;  4159          4262  2        ! once since that is all the syntax allows.
```

```
;4160     4263  2     !
;4161     4264  2     NOUN_NODE = 0;
;4162     4265  2     WHILE TRUE DO
;4163     4266  2         BEGIN
;4164     4267  2         WPTR = 0;
;4165     4268  2         KIND = DBG$K_DISP_NOKIND;
;4166     4269  2         IF .SET_FLAG THEN KIND = DBG$K_DISP_NORMAL;
;4167     4270  2         CONTENT_PTR = 0;
;4168     4271  2
;4169     4272  2
;4170     4273  2         ! Pick up the display name.
;4171     4274  2         !
;4172     4275  2         NAMEPTR = PARSE_DISPLAY_NAME(.INPUT_DESC, NOT .SET_FLAG);
;4173     4276  2
;4174     4277  2
;4175     4278  2         ! Pick up the window specification if one is present.  The window spec-
;4176     4279  2         ! ification may be either a parenthesized list of window parameters or
;4177     4280  2         ! an already declared window name.
;4178     4281  2         !
;4179     4282  2         IF DBG$NMATCH(.INPUT_DESC, DBG$CS_AT, 2)
;4180     4283  3         THEN
;4181     4284  4             BEGIN
;4182     4285  4
;4183     4286  4
;4184     4287  4             ! If the user is specifying a window specification of the form
;4185     4288  4             ! "(rbeg, rlen, cbeg, clen)", pick up the window parameters and
;4186     4289  4             ! create a Temporary Screen Window Entry for those parameters.
;4187     4290  4             !
;4188     4291  4             IF DBG$NMATCH(.INPUT_DESC, DBG$CS_LPAREN, 1)
;4189     4292  4             THEN
;4190     4293  5                 BEGIN
;4191     4294  5                 PARSE_WINDOW_PARAMETERS(.INPUT_DESC, RBEG, RLEN, CBEG, CLEN);
;4192     4295  5                 WPTR = DBG$SCR_CREATE_TEMP_WINDOW(.RBEG, .RLEN, .CBEG, .CLEN);
;4193     4296  5                 END
;4194     4297  5
;4195     4298  5
;4196     4299  5             ! Otherwise he must be specifying a window name.  Pick up that name,
;4197     4300  5             ! look it up the in the Screen Window List, and signal an error if
;4198     4301  5             ! there is no such window.
;4199     4302  5             !
;4200     4303  4             ELSE
;4201     4304  5                 BEGIN
;4202     4305  5                 WNAMEPTR = PARSE_WINDOW_NAME(.INPUT_DESC);
;4203     4306  5                 WPTR = DBG$SCR_LOOKUP_WINDOW(.WNAMEPTR);
;4204     4307  5                 IF .WPTR EQL 0 THEN SIGNAL(DBG$_NOSUCHWIND, 1, .WNAMEPTR);
;4205     4308  4                 END;
;4206     4309  4
;4207     4310  4             END
;4208     4311  4
;4209     4312  4
;4210     4313  4         ! If there is no window specification but this is a SET DISPLAY
;4211     4314  4         ! command, we want a default window specification.  This is done
;4212     4315  4         ! by creating a Temporary Screen Window Entry for window H1 or H2.
;4213     4316  4         ! Subsequent default windows alternate between H1 and H2.
;4214     4317  4         !
;4215     4318  3         ELSE IF .SET_FLAG
;4216     4319  3         THEN
```

```
;  4217          4320  4              BEGIN
;  4218          4321  4              WPTR = DBG$SCR_CREATE_TEMP_WINDOW(10*.DW_INDEX + 1, 9, 1, 132);
;  4219          4322  4              IF .DW_INDEX EQL 0 THEN DW_INDEX = 1 ELSE DW_INDEX = 0;
;  4220          4323  3              END;
;  4221          4324
;  4222          4325
;  4223          4326  3          ! Pick up the display kind if one is present.
;  4224          4327  3          !
;  4225          4328  3          SELECTONE TRUE OF
;  4226          4329  3              SET
;  4227          4330  3
;  4228          4331  3
;  4229          4332  3              ! Check for the NORMAL display kind.
;  4230          4333  3              !
;  4231          4334  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_NORMAL, 3)]:
;  4232          4335  3                  KIND = DBG$K_DISP_NORMAL;
;  4233          4336  3
;  4234          4337  3
;  4235          4338  3              ! Check for the DO(cmd-list) display kind.
;  4236          4339  3              !
;  4237          4340  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_DO, 2)]:
;  4238          4341  4                  BEGIN
;  4239          4342  4                  KIND = DBG$K_DISP_DO;
;  4240          4343  4                  IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_LPAREN, 1)
;  4241          4344  4                  THEN
;  4242          4345  4                      DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4243          4346  4
;  4244          4347  4                  DBG$NSAVE_BREAK_BUFFER(.INPUT_DESC, CONTENT_PTR, FALSE);
;  4245          4348  3                  END;
;  4246          4349  3
;  4247          4350  3
;  4248          4351  3              ! Check for the SOURCE o' SOURCE(cmd-list) display kind.  If the
;  4249          4352  3              ! (cmd-list) is missing, we leave CONTENT_PTR zeroed.
;  4250          4353  3              !
;  4251          4354  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_SOURCE, 3)]:
;  4252          4355  4                  BEGIN
;  4253          4356  4                  KIND = DBG$K_DISP_SOURCE;
;  4254          4357  4                  IF DBG$NMATCH(.INPUT_DESC, DBG$CS_LPAREN, 1)
;  4255          4358  4                  THEN
;  4256          4359  4                      DBG$NSAVE_BREAK_BUFFER(.INPUT_DESC, CONTENT_PTR, FALSE);
;  4257          4360  4
;  4258          4361  3                  END;
;  4259          4362  3
;  4260          4363  3
;  4261          4364  3              ! Check for the REGISTER display kind.
;  4262          4365  3              !
;  4263          4366  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_REGISTER, 3)]:
;  4264          4367  3                  KIND = DBG$K_DISP_REGISTER;
;  4265          4368  3
;  4266          4369  3              TES;
;  4267          4370  3
;  4268          4371  3
;  4269          4372  3          ! Now build a Noun Node for this display specification and its
;  4270          4373  3          ! various attributes.  Add that Noun Node to the linked list of
;  4271          4374  3          ! Noun Nodes attached to the Verb Node.
;  4272          4375  3          !
;  4273          4376  3          OLD_NOUN_PTR = .NOUN_NODE;
```

```
; 4274          4377  3          NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE_LONG);
; 4275          4378  3          NOUN_NODE[DBG$L_NOUN_VALUE] = .NAMEPTR;
; 4276          4379  3          NOUN_NODE[DBG$L_NOUN_VALUE2] = .WPTR;
; 4277          4380  3          NOUN_NODE[DBG$L_NOUN_VALUE3] = .KIND;
; 4278          4381  3          NOUN_NODE[DBG$L_NOUN_VALUE4] = .CONTENT_PTR;
; 4279          4382  3          IF .OLD_NOUN_PTR EQL 0
; 4280          4383  3          THEN
; 4281          4384  3              VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE
; 4282          4385  3
; 4283          4386  3          ELSE
; 4284          4387  3              OLD_NOUN_PTR[DBG$L_NOUN_LINK] = .NOUN_NODE;
; 4285          4388  3
; 4286          4389  3
; 4287          4390  3          ! If the next token is a carriage-return (end of command), we exit
; 4288          4391  3          ! the display specification loop.  However, if the next token is a
; 4289          4392  3          ! comma, we loop for the next display specification.
; 4290          4393  3          !
; 4291          4394  3          IF .INPUT_DESC[DSC$W_LENGTH] EQL 0 THEN EXITLOOP;
; 4292          4395  3          IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1) THEN EXITLOOP;
; 4293          4396  3          IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COMMA, 1)
; 4294          4397  3          THEN
; 4295          4398  3              DBG$SYNTAX_ERROR(.INPUT_DESC);
; 4296          4399  3
; 4297          4400  2          END;                                ! End of loop over display specs
; 4298          4401  2
; 4299          4402  2
; 4300          4403  2      ! The command processing is completed.  Now return.
; 4301          4404  2      !
; 4302          4405  2      RETURN;
; 4303          4406  2
; 4304          4407  1      END;

                                                    .PSECT  DBG$OWN,NOEXE,  PIC,2

                                00000000  01770 DW_INDEX:
                                                    .LONG   0                                                       ;

                                                    .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                    0FFC 00000      .ENTRY  DBG$SCR_PARSE_DISPLAY_CMD, Save R2,R3,R4,-              ; 4035
                                                            R5,R6,R7,R8,R9,R10,R11                                  ;
                          5E            1C  C2 00002      SUBL2   #28, SP                                         ;
                                        58  D4 00005      CLRL    HIDE_FLAG                                       ; 4114
                                        54  7C 00007      CLRQ    NOMARK_FLAG                                     ; 4116
                                        56  7C 00009      CLRQ    GENERATE_FLAG                                   ; 4120
                                        59  7C 0000B      CLRQ    CLEAR_FLAG                                      ; 4118
                                        5B  D4 0000D      CLRL    GENERATE_ALL_FLAG                               ; 4119
                                        7E  D4 0000F      CLRL    SIZE                                            ; 4121
                          52        04  AC  D0 00011      MOVL    INPUT_DESC, R2                                  ; 4127
                                        01  DD 00015 1$:   PUSHL   #1                                             ;
                              00000000' EF  9F 00017      PUSHAB  DBG$CS_SLASH                                    ;
                                        52  DD 0001D      PUSHL   R2                                              ;
                00000000G   00          03  FB 0001F      CALLS   #3, DBG$NMATCH                                  ;
```

```
                03              50  E8 00026          BLBS    R0, 2$                                  ;
                            015D 31  00029          BRW     24$                                     ;
                04        08    AC  E9 0002C 2$:      BLBC    SET_FLAG, 3$                            ; 4140
                                50  D4 00030          CLRL    R0                                      ;
                                11  11 00032          BRB     4$                                      ;
                                03  DD 00034 3$:      PUSHL   #3                                      ; 4141
                    00000000'   EF  9F 00036          PUSHAB  DBG$CS_REFRESH                          ;
                                52  DD 0003C          PUSHL   R2                                      ;
00000000G       00              03  FB 0003E          CALLS   #3, DBG$NMATCH                          ;
                01              50  D1 00045 4$:      CMPL    R0, #1                                  ; 4140
                                05  12 00048          BNEQ    6$                                      ;
                5A              01  D0 0004A          MOVL    #1, REFRESH_FLAG                        ; 4142
                                C6  11 0004D 5$:      BRB     1$                                      ;
                04        08    AC  E9 0004F 6$:      BLBC    SET_FLAG, 7$                            ; 4148
                                50  D4 00053          CLRL    R0                                      ;
                                11  11 00055          BRB     8$                                      ;
                                02  DD 00057 7$:      PUSHL   #2                                      ; 4149
                    00000000'   EF  9F 00059          PUSHAB  DBG$CS_CLEAR                            ;
                                52  DD 0005F          PUSHL   R2                                      ;
00000000G       00              03  FB 00061          CALLS   #3, DBG$NMATCH                          ;
                01              50  D1 00068 8$:      CMPL    R0, #1                                  ; 4148
                                05  12 0006B          BNEQ    10$                                     ;
                59              01  D0 0006D          MOVL    #1, CLEAR_FLAG                          ; 4150
                                A3  11 00070 9$:      BRB     1$                                      ;
                04        08    AC  E9 00072 10$:     BLBC    SET_FLAG, 11$                           ; 4156
                                50  D4 00076          CLRL    R0                                      ;
                                11  11 00078          BRB     12$                                     ;
                                02  DD 0007A 11$:     PUSHL   #2                                      ; 4157
                    00000000'   EF  9F 0007C          PUSHAB  DBG$CS_GENERATE                         ;
                                52  DD 00082          PUSHL   R2                                      ;
00000000G       00              03  FB 00084          CALLS   #3, DBG$NMATCH                          ;
                01              50  D1 0008B 12$:     CMPL    R0, #1                                  ; 4156
                                05  12 0008E          BNEQ    13$                                     ;
                56              01  D0 00090          MOVL    #1, GENERATE_FLAG                       ; 4158
                                80  11 00093          BRB     1$                                      ;
                                02  DD 00095 13$:     PUSHL   #2                                      ; 4163
                    00000000'   EF  9F 00097          PUSHAB  DBG$CS_HIDE                             ;
                                52  DD 0009D          PUSHL   R2                                      ;
00000000G       00              03  FB 0009F          CALLS   #3, DBG$NMATCH                          ;
                01              50  D1 000A6          CMPL    R0, #1                                  ;
                                05  12 000A9          BNEQ    14$                                     ;
                58              01  D0 000AB          MOVL    #1, HIDE_FLAG                           ; 4164
                                9D  11 000AE          BRB     5$                                      ;
                                02  DD 000B0 14$:     PUSHL   #2                                      ; 4169
                    00000000'   EF  9F 000B2          PUSHAB  DBG$CS_MARK_CHANGE                      ;
                                52  DD 000B8          PUSHL   R2                                      ;
00000000G       00              03  FB 000BA          CALLS   #3, DBG$NMATCH                          ;
                01              50  D1 000C1          CMPL    R0, #1                                  ;
                                07  12 000C4          BNEQ    15$                                     ;
                55              01  D0 000C6          MOVL    #1, MARK_FLAG                           ; 4171
                                54  D4 000C9          CLRL    NOMARK_FLAG                             ; 4172
                                80  11 000CB          BRB     5$                                      ; 4173
                04        08    AC  E9 000CD 15$:     BLBC    SET_FLAG, 16$                           ; 4179
                                50  D4 000D1          CLRL    R0                                      ;
                                11  11 000D3          BRB     17$                                     ;
                                04  DD 000D5 16$:     PUSHL   #4                                      ; 4187
                    00000000'   EF  9F 000D7          PUSHAB  DBG$CS_NOMARK_CHANGE                    ;
```

```
                        52  DD 000DD          PUSHL   R2                                                          ;
00000000G   00          03  FB 000DF          CALLS   #3, DBG$NMATCH                                              ;
            01          50  D1 000E6 17$:     CMPL    R0, #1                                                      ; 4179
                        05  12 000E9          BNEQ    18$                                                         ;
            54          01  7D 000EB          MOVQ    #1, NOMARK_FLAG                                             ; 4182
                        80  11 000EE          BRB     9$                                                          ; 4133
                        03  DD 000F0 18$:     PUSHL   #3                                                          ; 4189
                00000000' EF  9F 000F2        PUSHAB  DBG$CS_REMOVED                                              ;
                        52  DD 000F8          PUSHL   R2                                                          ;
00000000G   00          03  FB 000FA          CALLS   #3, DBG$NMATCH                                              ;
            01          50  D1 00101          CMPL    R0, #1                                                      ;
                        05  12 00104          BNEQ    19$                                                         ;
            57          01  D0 00106          MOVL    #1, REMOVED_FLAG                                            ; 4190
                        7B  11 00109          BRB     23$                                                         ;
                        02  DD 0010B 19$:     PUSHL   #2                                                          ; 4196
                00000000' EF  9F 0010D        PUSHAB  DBG$CS_SIZE                                                 ;
                        52  DD 00113          PUSHL   R2                                                          ;
00000000G   00          03  FB 00115          CALLS   #3, DBG$NMATCH                                              ;
            01          50  D1 0011C          CMPL    R0, #1                                                      ;
                        5C  12 0011F          BNEQ    22$                                                         ;
                        01  DD 00121          PUSHL   #1                                                          ; 4198
                00000000' EF  9F 00123        PUSHAB  DBG$CS_COLON                                                ;
                        52  DD 00129          PUSHL   R2                                                          ;
00000000G   00          03  FB 0012B          CALLS   #3, DBG$NMATCH                                              ;
            1D          50  E8 00132          BLBS    R0, 20$                                                     ;
                        01  DD 00135          PUSHL   #1                                                          ; 4201
                00000000' EF  9F 00137        PUSHAB  DBG$CS_EQUAL                                                ;
                        52  DD 0013D          PUSHL   R2                                                          ;
00000000G   00          03  FB 0013F          CALLS   #3, DBG$NMATCH                                              ;
            09          50  E8 00146          BLBS    R0, 20$                                                     ;
                        52  DD 00149          PUSHL   R2                                                          ; 4203
            00000000G   00  01  FB 0014B      CALLS   #1, DBG$SYNTAX_ERROR                                        ;
                    4004    8F  BB 00152 20$: PUSHR   #^M<R2,SP>                                                 ; 4207
            00000000G   00  02  FB 00156      CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER                               ;
                        6E  D5 0015D          TSTL    SIZE                                                        ; 4208
                        09  15 0015F          BLEQ    21$                                                         ;
            000003E8    8F  6E  D1 00161      CMPL    SIZE, #1000                                                 ;
                        1C  15 00168          BLEQ    23$                                                         ;
                        6E  DD 0016A 21$:     PUSHL   SIZE                                                        ; 4210
                        01  DD 0016C          PUSHL   #1                                                          ;
                00028A72    8F  DD 0016E      PUSHL   #166514                                                     ;
            00000000G   00  03  FB 00174      CALLS   #3, LIB$SIGNAL                                              ;
                        09  11 0017B          BRB     23$                                                         ; 4133
                        52  DD 0017D 22$:     PUSHL   R2                                                          ; 4219
            00000000G   00  01  FB 0017F      CALLS   #1, DBG$SYNTAX_ERROR                                        ;
                    FE8C    31  00186 23$:    BRW     1$                                                          ; 4127
                17          56  E9 00189 24$: BLBC    GENERATE_FLAG, 25$                                          ; 4230
                        01  DD 0018C          PUSHL   #1                                                          ;
                00000000' EF  9F 0018E        PUSHAB  DBG$CS_CR                                                   ;
                        52  DD 00194          PUSHL   R2                                                          ;
            00000000G   00  03  FB 00196      CALLS   #3, DBG$NMATCH                                              ;
                        03  50  E9 0019D      BLBC    R0, 25$                                                     ;
                        5B  01  D0 001A0      MOVL    #1, GENERATE_ALL_FLAG                                       ; 4232
                        53  D4 001A3 25$:     CLRL    FLAGWORD                                                    ; 4238
53          01          00  5A  F0 001A5      INSV    REFRESH_FLAG, #0, #1, FLAGWORD                              ; 4239
53          01          01  59  F0 001AA      INSV    CLEAR_FLAG, #1, #1, FLAGWORD                                ; 4240
53          01          02  5B  F0 001AF      INSV    GENERATE_ALL_FLAG, #2, #1, FLAGWORD                         ; 4241
```

```
                53  01          03          56  F0 001B4          INSV    GENERATE_FLAG, #3, #1, FLAGWORD          ; 4242
                53  01          04          58  F0 001B9          INSV    HIDE_FLAG, #4, #1, FLAGWORD              ; 4243
                53  01          05          55  F0 001BE          INSV    MARK_FLAG, #5, #1, FLAGWORD              ; 4244
                53  01          06          54  F0 001C3          INSV    NOMARK_FLAG, #6, #1, FLAGWORD            ; 4245
                53  01          07          57  F0 001C8          INSV    REMOVED_FLAG, #7, #1, FLAGWORD           ; 4246
                53  10          10          6E  F0 001CD          INSV    SIZE, #T6, #16, FLAGWORD                 ; 4247
                                            03  DD 001D2          PUSHL   #3                                       ; 4248
                        00000000G   00      01  FB 001D4          CALLS   #1, DBG$GET_TEMPMEM
                              04    A0      53  D0 001DB          MOVL    FLAGWORD, 4(ADVERB_NODE)                 ; 4249
                                    53  0C  AC  D0 001DF          MOVL    VERB_NODE, R3                            ; 4250
                              04    A3      50  D0 001E3          MOVL    ADVERB_NODE, 4(R3)
                                    01      5A  E9 001E7          BLBC    REFRESH_FLAG, 26$                        ; 4257
                                            04  001EA          RET
                                    01      5B  E9 001EB  26$:    BLBC    GENERATE_ALL_FLAG, 27$
                                            04  001EE          RET
                                            57  D4 001EF  27$:    CLRL    NOUN_NODE                                ; 4264
                                    55  08  AC  D2 001F1          MCOML   SET_FLAG, R5                             ; 4275
                                            54  D4 001F5  28$:    CLRL    WPTR                                     ; 4267
                                            5A  D4 001F7          CLRL    KIND                                     ; 4268
                                    03  08  AC  E9 001F9          BLBC    SET_FLAG, 29$                            ; 4269
                                    5A      01  D0 001FD          MOVL    #1, KIND
                                        14  AE  D4 00200  29$:    CLRL    CONTENT_PTR                              ; 4270
                                            24  BB 00203          PUSHR   #^M<R2,R5>                               ; 4275
                              0000V CF          02  FB 00205          CALLS   #2, PARSE_DISPLAY_NAME
                                    58          50  D0 0020A          MOVL    R0, NAMEPTR
                                            02  DD 0020D          PUSHL   #2                                       ; 4282
                                 00000000'  EF  9F 0020F          PUSHAB  DBG$CS_AT
                                            52  DD 00215          PUSHL   R2
                        00000000G   00      03  FB 00217          CALLS   #3, DBG$NMATCH
                                    66          50  E9 0021E          BLBC    R0, 31$
                                            01  DD 00221          PUSHL   #1                                       ; 4291
                                 00000000'  EF  9F 00223          PUSHAB  DBG$CS_LPAREN
                                            52  DD 00229          PUSHL   R2
                00000000G   00      03  FB 0022B          CALLS   #3, DBG$NMATCH
                                    29          50  E9 00232          BLBC    R0, 30$
                                        04  AE  9F 00235          PUSHAB  CLEN                                     ; 4294
                                        0C  AE  9F 00238          PUSHAB  CBEG
                                        14  AE  9F 0023B          PUSHAB  RLEN
                                        1C  AE  9F 0023E          PUSHAB  RBEG
                                            52  DD 00241          PUSHL   R2
                              0000V CF          05  FB 00243          CALLS   #5, PARSE_WINDOW_PARAMETERS
                                        04  AE  DD 00248          PUSHL   CLEN                                     ; 4295
                                        0C  AE  DD 0024B          PUSHL   CBEG
                                        14  AE  DD 0024E          PUSHL   RLEN
                                        1C  AE  DD 00251          PUSHL   RBEG
                              E482  CF          04  FB 00254          CALLS   #4, DBG$SCR_CREATE_TEMP_WINDOW
                                    54          50  D0 00259          MOVL    R0, WPTR
                                                5F  11 0025C          BRB     33$                                      ; 4291
                                            52  DD 0025E  30$:    PUSHL   R2                                       ; 4305
                              0000V CF          01  FB 00260          CALLS   #1, PARSE_WINDOW_NAME
                                    56          50  D0 00265          MOVL    R0, WNAMEPTR
                                            56  DD 00268          PUSHL   WNAMEPTR                                 ; 4306
                              F4D5  CF          01  FB 0026A          CALLS   #1, DBG$SCR_LOOKUP_WINDOW
                                    54          50  D0 0026F          MOVL    R0, WPTR
                                                49  12 00272          BNEQ    33$                                      ; 4307
                                            56  DD 00274          PUSHL   WNAMEPTR
                                            01  DD 00276          PUSHL   #1
```

```
                00028A6A  8F  DD 00278          PUSHL   #166506
    00000000G   00        03  FB 0027E          CALLS   #3, LIB$SIGNAL
                          36  11 00285          BRB     33$                         ;  4282
                32        08  AC  E9 00287 31$: BLBC    SET_FLAG, 33$               ;  4318
                7E        84  8F  9A 0028B      MOVZBL  #132, -(SP)                 ;  4321
                              01  DD 0028F      PUSHL   #1
                              09  DD 00291      PUSHL   #9
    50 00000000'  EF              0A  C5 00293  MULL3   #10, DW_INDEX, R0
                          01      A0  9F 0029B  PUSHAB  1(R0)
            E438  CF              04  FB 0029E  CALLS   #4, DBG$SCR_CREATE_TEMP_WINDOW
                  54              50  D0 002A3  MOVL    R0, WPTR
                      00000000'   EF  D5 002A6  TSTL    DW_INDEX                    ;  4322
                                  09  12 002AC  BNEQ    32$
    00000000'     EF              01  D0 002AE  MOVL    #1, DW_INDEX
                                  06  11 002B5  BRB     33$
                      00000000'   EF  D4 002B7 32$:     CLRL    DW_INDEX
                                  03  DD 002BD 33$:     PUSHL   #3                  ;  4334
                      00000000'   EF  9F 002BF      PUSHAB  DBG$CS_NORMAL
                                  52  DD 002C5      PUSHL   R2
    00000000G   00                03  FB 002C7      CALLS   #3, DBG$NMATCH
                01                50  D1 002CE      CMPL    R0, #1
                                  05  12 002D1      BNEQ    34$
                5A                01  D0 002D3      MOVL    #1, KIND                ;  4335
                                  73  11 002D6      BRB     37$
                                  02  DD 002D8 34$:     PUSHL   #2                  ;  4340
                      00000000'   EF  9F 002DA      PUSHAB  DBG$CS_DO
                                  52  DD 002E0      PUSHL   R2
    00000000G   00                03  FB 002E2      CALLS   #3, DBG$NMATCH
                01                50  D1 002E9      CMPL    R0, #1
                                  22  12 002EC      BNEQ    35$
                5A                02  D0 002EE      MOVL    #2, KIND                ;  4342
                                  01  DD 002F1      PUSHL   #1                      ;  4343
                      00000000'   EF  9F 002F3      PUSHAB  DBG$CS_LPAREN
                                  52  DD 002F9      PUSHL   R2
    00000000G   00                03  FB 002FB      CALLS   #3, DBG$NMATCH
                38                50  E8 00302      BLBS    R0, 36$
                                  52  DD 00305      PUSHL   R2                      ;  4345
    00000000G   00                01  FB 00307      CALLS   #1, DBG$SYNTAX_ERROR
                                  2D  11 0030E      BRB     36$                     ;  4347
                                  03  DD 00310 35$:     PUSHL   #3                  ;  4354
                      00000000'   EF  9F 00312      PUSHAB  DBG$CS_SOURCE
                                  52  DD 00318      PUSHL   R2
    00000000G   00                03  FB 0031A      CALLS   #3, DBG$NMATCH
                01                50  D1 00321      CMPL    R0, #1
                                  27  12 00324      BNEQ    38$
                5A                03  D0 00326      MOVL    #3, KIND                ;  4356
                                  01  DD 00329      PUSHL   #1                      ;  4357
                      00000000'   EF  9F 0032B      PUSHAB  DBG$CS_LPAREN
                                  52  DD 00331      PUSHL   R2
    00000000G   00                03  FB 00333      CALLS   #3, DBG$NMATCH
                29                50  E9 0033A      BLBC    R0, 39$
                                  7E  D4 0033D 36$:     CLRL    -(SP)               ;  4359
                            18    AE  9F 0033F      PUSHAB  CONTENT_PTR
                                  52  DD 00342      PUSHL   R2
    00000000G   00                03  FB 00344      CALLS   #3, DBG$NSAVE_BREAK_BUFFER
                                  19  11 0034B 37$:     BRB     39$                 ;  4328
                                  03  DD 0034D 38$:     PUSHL   #3                  ;  4366
```

```
                    00000000'  EF  9F 0034F          PUSHAB  DBG$CS_REGISTER                          ;
                               52  DD 00355          PUSHL   R2                                      ;
00000000G             00       03  FB 00357          CALLS   #3, DBG$NMATCH                          ;
                      01       50  D1 0035E          CMPL    R0, #1                                  ;
                               03  12 00361          BNEQ    39$                                     ;
                      5A       04  D0 00363          MOVL    #4, KIND                                ; 4367
                      59       57  D0 00366 39$:     MOVL    NOUN_NODE, OLD_NOUN_PTR                 ; 4376
                               06  DD 00369          PUSHL   #6                                      ; 4377
00000000G             00       01  FB 0036B          CALLS   #1, DBG$GET_TEMPMEM                     ;
                      57       50  D0 00372          MOVL    R0, NOUN_NODE                           ;
                      67       58  D0 00375          MOVL    NAMEPTR, (NOUN_NODE)                    ; 4378
                   0C A7       54  D0 00378          MOVL    WPTR, 12(NOUN_NODE)                     ; 4379
                   10 A7       5A  D0 0037C          MOVL    KIND, 16(NOUN_NODE)                     ; 4380
                   14 A7  14   AE  D0 00380          MOVL    CONTENT_PTR, 20(NOUN_NODE)              ; 4381
                               59  D5 00385          TSTL    OLD_NOUN_PTR                            ; 4382
                               06  12 00387          BNEQ    40$                                     ;
                   08 A3       57  D0 00389          MOVL    NOUN_NODE, 8(R3)                        ; 4384
                               04  11 0038D          BRB     41$                                     ;
                   08 A9       57  D0 0038F 40$:     MOVL    NOUN_NODE, 8(OLD_NOUN_PTR)              ; 4387
                               62  B5 00393 41$:     TSTW    (R2)                                    ; 4394
                               34  13 00395          BEQL    43$                                     ;
                               01  DD 00397          PUSHL   #1                                      ; 4395
                    00000000'  EF  9F 00399          PUSHAB  DBG$CS_CR                               ;
                               52  DD 0039F          PUSHL   R2                                      ;
00000000G             00       03  FB 003A1          CALLS   #3, DBG$NMATCH                          ;
                      20       50  E8 003A8          BLBS    R0, 43$                                 ;
                               01  DD 003AB          PUSHL   #1                                      ; 4396
                    00000000'  EF  9F 003AD          PUSHAB  DBG$CS_COMMA                            ;
                               52  DD 003B3          PUSHL   R2                                      ;
00000000G             00       03  FB 003B5          CALLS   #3, DBG$NMATCH                          ;
                      09       50  E8 003BC          BLBS    R0, 42$                                 ;
                               52  DD 003BF          PUSHL   R2                                      ; 4398
00000000G             00       01  FB 003C1          CALLS   #1, DBG$SYNTAX_ERROR                    ;
                         FE2A  31  03 003C8 42$:     BRW     28$                                     ; 4265
                               04  03 003CB 43$:     RET                                             ; 4407

; Routine Size:  972 bytes,    Routine Base:  DBG$CODE + 2432
```

```
;  4306          4408  1 GLOBAL ROUTINE DBG$SCR_PARSE_SAVE_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  4307          4409  1 !
;  4308          4410  1 ! FUNCTION
;  4309          4411  1 !       This routine parses the SAVE command.  It accepts a command line
;  4310          4412  1 !       string descriptor as input and produces a Verb Node for the parsed
;  4311          4413  1 !       SAVE command as output.  That Verb Node and its attached Noun Node
;  4312          4414  1 !       list later serve as input to DBG$SCR_EXECUTE_SAVE_CMD which actually
;  4313          4415  1 !       executes the command.
;  4314          4416  1 !
;  4315          4417  1 ! INPUTS
;  4316          4418  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  4317          4419  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  4318          4420  1 !                    first character after the SAVE keyword.
;  4319          4421  1 !
;  4320          4422  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  4321          4423  1 !                   being parsed.
;  4322          4424  1 !
;  4323          4425  1 ! OUTPUTS
;  4324          4426  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  4325          4427  1 !                    first character after the end of the command.  This normally
;  4326          4428  1 !                    means that the input string is exhausted.
;  4327          4429  1 !
;  4328          4430  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it and its
;  4329          4431  1 !                   attached Noun Nodes contains all information picked up
;  4330          4432  1 !                   during the parse of the SAVE command.
;  4331          4433  1 !
;  4332          4434  1
;  4333          4435  2     BEGIN
;  4334          4436  2
;  4335          4437  2     MAP
;  4336          4438  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  4337          4439  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  4338          4440  2
;  4339          4441  2     LOCAL
;  4340          4442  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;  4341          4443  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to current display name
;  4342          4444  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Pointer to the current Noun Node
;  4343          4445  2         OLD_NOUN_PTR: REF DBG$NOUN_NODE;! Temporary Noun Node pointer
;  4344          4446  2
;  4345          4447  2
;  4346          4448  2
;  4347          4449  2     ! Loop to pick up all the display SAVE specifications.  Each such specifi-
;  4348          4450  2     ! cation is of the form "disp1 AS disp2", where disp1 is the existing dis-
;  4349          4451  2     ! play to be saved and disp2 is the new display to be created with the same
;  4350          4452  2     ! contents as disp1.  The SAVE specifications are separated by commas.
;  4351          4453  2     !
;  4352          4454  2     NOUN_NODE = 0;
;  4353          4455  2     WHILE TRUE DO
;  4354          4456  3         BEGIN
;  4355          4457  3
;  4356          4458  3
;  4357          4459  3         ! Pick up the name of the existing display to be saved (disp1) and make
;  4358          4460  3         ! sure there is such a display.
;  4359          4461  3         !
;  4360          4462  3         NAMEPTR = PARSE_DISPLAY_NAME(.INPUT_DESC, TRUE);
;  4361          4463  3         DISPTR = DBG$SCR_LOOKUP_DISPLAY(.NAMEPTR);
;  4362          4464  3         IF .DISPTR EQL 0 THEN SIGNAL(DBG$_NOSUCHDISP, 1, .NAMEPTR);
```

```
;  4363          4465  
;  4364          4466  
;  4365          4467          ! Scan past the AS keyword and pick up the name of the new display
;  4366          4468          ! to be created (disp2).
;  4367          4469          !
;  4368          4470          IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_AS, 2)
;  4369          4471          THEN
;  4370          4472              DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4371          4473
;  4372          4474          NAMEPTR = PARSE_DISPLAY_NAME(.INPUT_DESC, FALSE);
;  4373          4475
;  4374          4476
;  4375          4477          ! Allocate and build a Noun Node for this SAVE specification.  Then
;  4376          4478          ! add it to the Verb Node's Noun Node chain.
;  4377          4479          !
;  4378          4480          OLD_NOUN_PTR = .NOUN_NODE;
;  4379          4481          NOUN_NODE = DBG$GET_MEMORY(DBG$K_NOUN_NODE_SIZE);
;  4380          4482          NOUN_NODE[DBG$L_NOUN_VALUE] = .DISPTR;
;  4381          4483          NOUN_NODE[DBG$L_NOUN_VALUE2] = .NAMEPTR;
;  4382          4484          IF .OLD_NOUN_PTR EQL 0
;  4383          4485          THEN
;  4384          4486              VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE
;  4385          4487
;  4386          4488          ELSE
;  4387          4489              OLD_NOUN_PTR[DBG$L_NOUN_LINK] = .NOUN_NODE;
;  4388          4490
;  4389          4491
;  4390          4492          ! If we find a comma next, we loop for the next SAVE specification.
;  4391          4493          ! Otherwise, we exit the SAVE specification loop.
;  4392          4494          !
;  4393          4495          IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COMMA, 1) THEN EXITLOOP;
;  4394          4496
;  4395          4497      2   END;                                ! End of loop over SAVE specifications
;  4396          4498      2
;  4397          4499      2
;  4398          4500      2   ! The SAVE command is fully and successfully parsed.  Now return.
;  4399          4501      2   !
;  4400          4502      2   RETURN;
;  4401          4503      2
;  4402          4504      1   END;


                                    00FC 00000          .ENTRY  DBG$SCR_PARSE_SAVE_CMD, Save R2,R3,R4,R5,-      ; 4408
                                                                R6,R7
                        57 00000000G    00   9E 00002          MOVAB   DBG$NMATCH, R7
                                        52   D4 00009          CLRL    NOUN_NODE                                  ; 4454
                        53        04    AC   D0 0000B          MOVL    INPUT_DESC, R3                             ; 4462
                                        01   DD 0000F  1$:      PUSHL   #1
                                        53   DD 00011          PUSHL   R3
                  0000V CF              02   FB 00013          CALLS   #2, PARSE_DISPLAY_NAME
                        55              50   D0 00018          MOVL    R0, NAMEPTR
                                        55   DD 0001B          PUSHL   NAMEPTR                                    ; 4463
                  F20C  CF              01   FB 0001D          CALLS   #1, DBG$SCR_LOOKUP_DISPLAY
                        56              50   D0 00022          MOVL    R0, DISPTR
```

```
                                11  12 00025          BNEQ    2$                                      ; 4464
                                55  DD 00027          PUSHL   NAMEPTR                                 ;
                                01  DD 00029          PUSHL   #1                                      ;
                    00028A62    8F  DD 0002B          PUSHL   #166498                                 ;
00000000G   00                  03  FB 00031          CALLS   #3, LIB$SIGNAL                          ;
                                02  DD 00038 2$:      PUSHL   #2                                      ; 4470
                    00000000'   EF  9F 0003A          PUSHAB  DBG$CS_AS                               ;
                                53  DD 00040          PUSHL   R3                                      ;
                67              03  FB 00042          CALLS   #3, DBG$NMATCH                          ;
                09              50  E8 00045          BLBS    R0, 3$                                  ;
                                53  DD 00048          PUSHL   R3                                      ; 4472
00000000G   00                  01  FB 0004A          CALLS   #1, DBG$SYNTAX_ERROR                    ;
                                7E  D4 00051 3$:      CLRL    -(SP)                                   ; 4474
                                53  DD 00053          PUSHL   R3                                      ;
     0000V      CF              02  FB 00055          CALLS   #2, PARSE_DISPLAY_NAME                  ;
                55              50  D0 0005A          MOVL    R0, NAMEPTR                             ;
                54              52  D0 0005D          MOVL    NOUN_NODE, OLD_NOUN_PTR                 ; 4480
                                04  DD 00060          PUSHL   #4                                      ; 4481
00000000G   00                  01  FB 00062          CALLS   #1, DBG$GET_MEMORY                      ;
                52              50  D0 00069          MOVL    R0, NOUN_NODE                           ;
                62              56  D0 0006C          MOVL    DISPTR, (NOUN_NODE)                     ; 4482
          0C    A2              55  D0 0006F          MOVL    NAMEPTR, 12(NOUN_NODE)                  ; 4483
                                54  D5 00073          TSTL    OLD_NOUN_PTR                            ; 4484
                                0A  12 00075          BNEQ    4$                                      ;
                50      08      AC  D0 00077          MOVL    VERB_NODE, R0                           ; 4486
          08    A0              52  D0 0007B          MOVL    NOUN_NODE, 8(R0)                        ;
                                04  11 0007F          BRB     5$                                      ;
          08    A4              52  D0 00081 4$:      MOVL    NOUN_NODE, 8(OLD_NOUN_PTR)              ; 4489
                                01  DD 00085 5$:      PUSHL   #1                                      ; 4495
                    00000000'   EF  9F 00087          PUSHAB  DBG$CS_COMMA                            ;
                                53  DD 0008D          PUSHL   R3                                      ;
                67              03  FB 0008F          CALLS   #3, DBG$NMATCH                          ;
                03              50  E9 00092          BLBC    R0, 6$                                  ;
                            FF77    31 00095          BRW     1$                                      ;
                                04  00098 6$:        RET                                             ; 4504

; Routine Size:  153 bytes,    Routine Base:  DBG$CODE + 27FE
```

```
;  4404          4505  1 GLOBAL ROUTINE DBG$SCR_PARSE_SCROLL_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  4405          4506  1 !
;  4406          4507  1 ! FUNCTION
;  4407          4508  1 !       This routine parses the SCROLL command. It accepts a command line
;  4408          4509  1 !       string descriptor as input and produces a Verb Node for the parsed
;  4409          4510  1 !       SCROLL command as output.  That Verb Node and its attached Noun
;  4410          4511  1 !       Node later serves as input to DBG$SCR_EXECUTE_SCROLL_CMD which
;  4411          4512  1 !       actually executes the command.
;  4412          4513  1 !
;  4413          4514  1 ! INPUTS
;  4414          4515  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  4415          4516  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  4416          4517  1 !                    first character after the SCROLL keyword.                  ;
;  4417          4518  1 !
;  4418          4519  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  4419          4520  1 !                   being parsed.
;  4420          4521  1 !
;  4421          4522  1 ! OUTPUTS
;  4422          4523  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  4423          4524  1 !                    first character after the end of the command.  This normally
;  4424          4525  1 !                    means that the input string is exhausted.
;  4425          4526  1 !
;  4426          4527  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it and its
;  4427          4528  1 !                   attached Noun Node contains all information picked up
;  4428          4529  1 !                   during the parse of the SCROLL command.
;  4429          4530  1 !
;  4430          4531  1
;  4431          4532  2     BEGIN
;  4432          4533  2
;  4433          4534  2     MAP
;  4434          4535  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  4435          4536  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  4436          4537  2
;  4437          4538  2     LOCAL
;  4438          4539  2         AMOUNT,                         ! Amount to scroll specified display
;  4439          4540  2         DIRECTION,                      ! Direction to scroll specified display
;  4440          4541  2         DISPID: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;  4441          4542  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to ASCIC name of display
;  4442          4543  2         NOUN_NODE: REF DBG$NOUN_NODE;   ! Pointer to Noun Node
;  4443          4544  2
;  4444          4545  2
;  4445          4546  2
;  4446          4547  2     ! Initialize the default direction and scrolling amount.
;  4447          4548  2     !
;  4448          4549  2     DIRECTION = DBG$K_SCROLL_DOWN;
;  4449          4550  2     AMOUNT = -1;
;  4450          4551  2
;  4451          4552  2
;  4452          4553  2     ! Parse the SCROLL command qualifier (if any).  The qualifier has the form
;  4453          4554  2     ! /direction[:amount] or /TOP or /BOTTOM.  The qualifier thus indicates
;  4454          4555  2     ! the scrolling direction and amount.
;  4455          4556  2     !
;  4456          4557  2     IF DBG$NMATCH(.INPUT_DESC, DBG$CS_SLASH, 1)
;  4457          4558  2     THEN
;  4458          4559  3         BEGIN
;  4459          4560  3
;  4460          4561  3
```

```
;  4461          4562  3          ! Pick up the qualifier name (the scrolling direction).
;  4462          4563  3          !
;  4463          4564  3          SELECTONE TRUE OF
;  4464          4565  3              SET
;  4465          4566  3
;  4466          4567  3
;  4467          4568  3              ! Check for SCROLL/UP:n qualifier.
;  4468          4569  3              !
;  4469          4570  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_UP, 1)]:
;  4470          4571  3                  DIRECTION = DBG$K_SCROLL_UP;
;  4471          4572  3
;  4472          4573  3
;  4473          4574  3              ! Check for SCROLL/DOWN:n qualifier.
;  4474          4575  3              !
;  4475          4576  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_DOWN, 1)]:
;  4476          4577  3                  DIRECTION = DBG$K_SCROLL_DOWN;
;  4477          4578  3
;  4478          4579  3
;  4479          4580  3              ! Check for SCROLL/LEFT:n qualifier.
;  4480          4581  3              !
;  4481          4582  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_LEFT, 1)]:
;  4482          4583  3                  DIRECTION = DBG$K_SCROLL_LEFT;
;  4483          4584  3
;  4484          4585  3
;  4485          4586  3              ! Check for SCROLL/RIGHT:n qualifier.
;  4486          4587  3              !
;  4487          4588  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_RIGHT, 1)]:
;  4488          4589  3                  DIRECTION = DBG$K_SCROLL_RIGHT;
;  4489          4590  3
;  4490          4591  3
;  4491          4592  3              ! Check for SCROLL/TOP qualifier.
;  4492          4593  3              !
;  4493          4594  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_TOP, 1)]:
;  4494          4595  4                  BEGIN
;  4495          4596  4                  DIRECTION = DBG$K_SCROLL_UP;
;  4496          4597  4                  AMOUNT = 65001;
;  4497          4598  3                  END;
;  4498          4599  3
;  4499          4600  3
;  4500          4601  3              ! Check for SCROLL/BOTTOM qualifier.
;  4501          4602  3              !
;  4502          4603  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_BOTTOM, 1)]:
;  4503          4604  4                  BEGIN
;  4504          4605  4                  DIRECTION = DBG$K_SCROLL_DOWN;
;  4505          4606  4                  AMOUNT = 65001;
;  4506          4607  3                  END;
;  4507          4608  3
;  4508          4609  3
;  4509          4610  3              ! If the qualifier is none of the above, we have a syntax error.
;  4510          4611  3              !
;  4511          4612  3              [OTHERWISE]:
;  4512          4613  3                  DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4513          4614  3
;  4514          4615  3              TES;
;  4515          4616  3
;  4516          4617  3
;  4517          4618  3          ! For all qualifiers except /TOP and /BOTTOM we now pick up the
```

```
; 4518          4619  3          ! scrolling amount if it is specified.
; 4519          4620  3          !
; 4520          4621  3          IF .AMOUNT EQL -1
; 4521          4622  3          THEN
; 4522          4623  4              BEGIN
; 4523          4624  5              IF (IF DBG$NMATCH(.INPUT_DESC, DBG$CS_COLON, 1) THEN TRUE
; 4524          4625  5                  ELSE DBG$NMATCH(.INPUT_DESC, DBG$CS_EQUAL, 1))
; 4525          4626  4              THEN
; 4526          4627  5                  BEGIN
; 4527          4628  5                  DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, AMOUNT);
; 4528          4629  5                  IF .AMOUNT GTR 65000 THEN AMOUNT = 65000;
; 4529          4630  4                  END;
; 4530          4631  4
; 4531          4632  3              END;
; 4532          4633  3
; 4533          4634  2          END;                            ! End of code to pick up qualifier
; 4534          4635  2
; 4535          4636  2
; 4536          4637  2      ! Now pick up the name of the display to be scrolled.  If no display name
; 4537          4638  2      ! is specified, use the default scrolling display.  If a name is specified,
; 4538          4639  2      ! convert that display name to a Display ID and make sure that such a
; 4539          4640  2      ! display exists.
; 4540          4641  2      !
; 4541          4642  2      IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1)
; 4542          4643  2      THEN
; 4543          4644  3          BEGIN
; 4544          4645  3          DISPID = .DBG$SCR_CURDISP_SCROLL;
; 4545          4646  3          IF .DISPID EQL 0 THEN SIGNAL(DBG$_NOSCROLL);
; 4546          4647  3          END
; 4547          4648  3
; 4548          4649  2      ELSE
; 4549          4650  3          BEGIN
; 4550          4651  3          NAMEPTR = PARSE_DISPLAY_NAME(.INPUT_DESC, TRUE);
; 4551          4652  3          DISPID = DBG$SCR_LOOKUP_DISPLAY(.NAMEPTR);
; 4552          4653  3          IF .DISPID EQL 0 THEN SIGNAL(DBG$_NOSUCHDISP, 1, .NAMEPTR);
; 4553          4654  2          END;
; 4554          4655  2
; 4555          4656  2
; 4556          4657  2      ! If no scrolling amount was specified, we determine the actual scrolling
; 4557          4658  2      ! amount based on the scrolling direction and the size of the display.
; 4558          4659  2      !
; 4559          4660  2      IF .AMOUNT EQL -1
; 4560          4661  2      THEN
; 4561          4662  3          BEGIN
; 4562          4663  3          IF (.DIRECTION EQL DBG$K_SCROLL_UP) OR
; 4563          4664  4             (.DIRECTION EQL DBG$K_SCROLL_DOWN)
; 4564          4665  3          THEN
; 4565          4666  3              AMOUNT = .DISPID[DBG$W_DISP_RLEN] - (.DISPID[DBG$W_DISP_RLEN]-1)/4
; 4566          4667  3
; 4567          4668  3          ELSE
; 4568          4669  3              AMOUNT = 8;
; 4569          4670  3
; 4570          4671  2          END;
; 4571          4672  2
; 4572          4673  2
; 4573          4674  2      ! Now build the Noun Node to hold the scrolling direction and amount and
; 4574          4675  2      ! the specified Display ID.  Link it to the Verb Node and then return.
```

```
;  4575          4676  2     !
;  4576          4677  2     NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE_LONG);
;  4577          4678  2     NOUN_NODE[DBG$L_NOUN_VALUE] = .DISPID;
;  4578          4679  2     NOUN_NODE[DBG$L_NOUN_VALUE2] = .DIRECTION;
;  4579          4680  2     NOUN_NODE[DBG$L_NOUN_VALUE3] = .AMOUNT;
;  4580          4681  2     VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;
;  4581          4682  2     RETURN;
;  4582          4683  2
;  4583          4684  1     END;


                                00FC 00000          .ENTRY  DBG$SCR_PARSE_SCROLL_CMD, Save R2,R3,R4,R5,-  ; 4505
                                                            R6,R7                                            ;
                57 00000000G    00   9E 00002          MOVAB   LIB$SIGNAL, R7                               ;
                56 00000000'    EF   9E 00009          MOVAB   DBG$CS_SLASH, R6                             ;
                55 00000000G    00   9E 00010          MOVAB   DBG$NMATCH, R5                               ;
                54              02   D0 00017          MOVL    #2, DIRECTION                                ; 4549
                7E              01   CE 0001A          MNEGL   #1, AMOUNT                                   ; 4550
                                01   DD 0001D          PUSHL   #1                                           ; 4557
                                56   DD 0001F          PUSHL   R6                                           ;
                53        04    AC   D0 00021          MOVL    INPUT_DESC, R3                               ;
                                53   DD 00025          PUSHL   R3                                           ;
                65              03   FB 00027          CALLS   #3, DBG$NMATCH                               ;
                03              50   E8 0002A          BLBS    R0, 1$                                       ;
                              00C5   31 0002D          BRW     11$                                          ;
                                01   DD 00030 1$:      PUSHL   #1                                           ; 4570
                          0D    A6   9F 00032          PUSHAB  DBG$CS_UP                                    ;
                                53   DD 00035          PUSHL   R3                                           ;
                65              03   FB 00037          CALLS   #3, DBG$NMATCH                               ;
                01              50   D1 0003A          CMPL    R0, #1                                       ;
                                05   12 0003D          BNEQ    2$                                           ;
                54              01   D0 0003F          MOVL    #1, DIRECTION                                ; 4571
                                74   11 00042          BRB     9$                                           ;
                                01   DD 00044 2$:      PUSHL   #1                                           ; 4576
                        FF7E    C6   9F 00046          PUSHAB  DBG$CS_DOWN                                  ;
                                53   DD 0004A          PUSHL   R3                                           ;
                65              03   FB 0004C          CALLS   #3, DBG$NMATCH                               ;
                01              50   D1 0004F          CMPL    R0, #1                                       ;
                                05   12 00052          BNEQ    3$                                           ;
                54              02   D0 00054          MOVL    #2, DIRECTION                                ; 4577
                                5F   11 00057          BRB     9$                                           ;
                                01   DD 00059 3$:      PUSHL   #1                                           ; 4582
                          A1    A6   9F 0005B          PUSHAB  DBG$CS_LEFT                                  ;
                                53   DD 0005E          PUSHL   R3                                           ;
                65              03   FB 00060          CALLS   #3, DBG$NMATCH                               ;
                01              50   D1 00063          CMPL    R0, #1                                       ;
                                05   12 00066          BNEQ    4$                                           ;
                54              03   D0 00068          MOVL    #3, DIRECTION                                ; 4583
                                4B   11 0006B          BRB     9$                                           ;
                                01   DD 0006D 4$:      PUSHL   #1                                           ; 4588
                          E9    A6   9F 0006F          PUSHAB  DBG$CS_RIGHT                                 ;
                                53   DD 00072          PUSHL   R3                                           ;
                65              03   FB 00074          CALLS   #3, DBG$NMATCH                               ;
                01              50   D1 00077          CMPL    R0, #1                                       ;
```

```
                                05  12 0007A          BNEQ    5$                                  ;
                          54    04  D0 0007C          MOVL    #4, DIRECTION                       ; 4589
                                37  11 0007F          BRB     9$                                  ;
                                01  DD 00081 5$:      PUSHL   #1                                  ; 4594
                             09 A6  9F 00083          PUSHAB  DBG$CS_TOP                          ;
                                53  DD 00086          PUSHL   R3                                  ;
                          65    03  FB 00088          CALLS   #3, DBG$NMATCH                      ;
                          01    50  D1 0008B          CMPL    R0, #1                              ;
                                05  12 0008E          BNEQ    6$                                  ;
                          54    01  D0 00090          MOVL    #1, DIRECTION                       ; 4596
                                13  11 00093          BRB     7$                                  ; 4597
                                01  DD 00095 6$:      PUSHL   #1                                  ; 4603
                          FF68  C6  9F 00097          PUSHAB  DBG$CS_BOTTOM                       ;
                                53  DD 0009B          PUSHL   R3                                  ;
                          65    03  FB 0009D          CALLS   #3, DBG$NMATCH                      ;
                          01    50  D1 000A0          CMPL    R0, #1                              ;
                                0A  12 000A3          BNEQ    8$                                  ;
                          54    02  D0 000A5          MOVL    #2, DIRECTION                       ; 4605
                          6E FDE9  8F 3C 000A8 7$:    MOVZWL  #65001, AMOUNT                      ; 4606
                                09  11 000AD          BRB     9$                                  ; 4564
                                53  DD 000AF 8$:      PUSHL   R3                                  ; 4613
                00000000G 00    01  FB 000B1          CALLS   #1, DBG$SYNTAX_ERROR                ;
                FFFFFFFF  8F    6E  D1 000B8 9$:      CMPL    AMOUNT, #-1                         ; 4621
                                34  12 000BF          BNEQ    11$                                 ;
                                01  DD 000C1          PUSHL   #1                                  ; 4624
                          FF75  C6  9F 000C3          PUSHAB  DBG$CS_COLON                        ;
                                53  DD 000C7          PUSHL   R3                                  ;
                          65    03  FB 000C9          CALLS   #3, DBG$NMATCH                      ;
                          0D    50  E8 000CC          BLBS    R0, 10$                             ;
                                01  DD 000CF          PUSHL   #1                                  ; 4625
                             83 A6  9F 000D1          PUSHAB  DBG$CS_EQUAL                        ;
                                53  DD 000D4          PUSHL   R3                                  ;
                          65    03  FB 000D6          CALLS   #3, DBG$NMATCH                      ;
                          19    50  E9 000D9          BLBC    R0, 11$                             ;
                          4008  8F  BB 000DC 10$:     PUSHR   #^M<R3,SP>                          ; 4628
                00000000G 00    02  FB 000E0          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER       ;
                0000FDE8  8F    6E  D1 000E7          CMPL    AMOUNT, #65000                      ; 4629
                                05  15 000EE          BLEQ    11$                                 ;
                          6E FDE8  8F 3C 000F0        MOVZWL  #65000, AMOUNT                      ;
                                01  DD 000F5 11$:     PUSHL   #1                                  ; 4642
                          FF79  C6  9F 000F7          PUSHAB  DBG$CS_CR                           ;
                                53  DD 000FB          PUSHL   R3                                  ;
                          65    03  FB 000FD          CALLS   #3, DBG$NMATCH                      ;
                          14    50  E9 00100          BLBC    R0, 12$                             ;
                          52 00000000'  EF  D0 00103  MOVL    DBG$SCR_CURDISP_SCROLL, DISPID      ; 4645
                                30  12 0010A          BNEQ    13$                                 ; 4646
                             00028A7A 8F  DD 0010C    PUSHL   #166522                             ;
                          67    01  FB 00112          CALLS   #1, LIB$SIGNAL                      ;
                                25  11 00115          BRB     13$                                 ; 4642
                                01  DD 00117 12$:     PUSHL   #1                                  ; 4651
                                53  DD 00119          PUSHL   R3                                  ;
                    0000V       CF  02  FB 0011B      CALLS   #2, PARSE_DISPLAY_NAME              ;
                          53    50  D0 00120          MOVL    R0, NAMEPTR                         ;
                                53  DD 00123          PUSHL   NAMEPTR                             ; 4652
                    F06B        CF  01  FB 00125      CALLS   #1, DBG$SCR_LOOKUP_DISPLAY          ;
                          52    50  D0 0012A          MOVL    R0, DISPID                          ;
                                0D  12 0012D          BNEQ    13$                                 ; 4653
```

```
                                53  DD 0012F          PUSHL   NAMEPTR                                    ;
                                01  DD 00131          PUSHL   #1                                        ;
                    00028A62    8F  DD 00133          PUSHL   #166498                                   ;
                          67        03  FB 00139          CALLS   #3, LIB$SIGNAL                        ;
            FFFFFFFF      8F        6E  D1 0013C 13$:     CMPL    AMOUNT, #-1                           ; 4660
                                    20  12 00143          BNEQ    16$                                   ;
                          01        54  D1 00145          CMPL    DIRECTION, #1                         ; 4663
                                    05  13 00148          BEQL    14$                                   ;
                          02        54  D1 0014A          CMPL    DIRECTION, #2                         ; 4664
                                    13  12 0014D          BNEQ    15$                                   ;
                          50    12  A2  3C 0014F 14$:     MOVZWL  18(DISPID), R0                        ; 4666
                                    50  D7 00153          DECL    R0                                    ;
                          50        04  C6 00155          DIVL2   #4, R0                                ;
                          51    12  A2  3C 00158          MOVZWL  18(DISPID), R1                        ;
                6E        51        50  C3 0015C          SUBL3   R0, R1, AMOUNT                        ;
                                    03  11 00160          BRB     16$                                   ;
                          6E        08  D0 00162 15$:     MOVL    #8, AMOUNT                            ; 4669
                                    06  DD 00165 16$:     PUSHL   #6                                    ; 4677
            00000000G     00        01  FB 00167          CALLS   #1, DBG$GET_TEMPMEM                   ;
                          60        52  D0 0016E          MOVL    DISPID, (NOUN_NODE)                   ; 4678
                    0C    A0        54  D0 00171          MOVL    DIRECTION, 12(NOUN_NODE)              ; 4679
                    10    A0        6E  D0 00175          MOVL    AMOUNT, 16(NOUN_NODE)                 ; 4680
                          51    08  AC  D0 00179          MOVL    VERB_NODE, R1                         ; 4681
                    08    A1        50  D0 0017D          MOVL    NOUN_NODE, 8(R1)                      ;
                                    04  00181          RET                                           ; 4684

; Routine Size:  386 bytes,    Routine Base:  DBG$CODE + 2897
```

```
;  4585          4685  1 GLOBAL ROUTINE DBG$SCR_PARSE_SELECT_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  4586          4686  1 !
;  4587          4687  1 ! FUNCTION
;  4588          4688  1 !       This routine parses the SELECT command. It accepts a command line
;  4589          4689  1 !       string descriptor as input and produces a verb node for the parsed
;  4590          4690  1 !       SELECT command as output.  That verb node and its attached noun node
;  4591          4691  1 !       later serve as input to EXECUTE_SELECT_CMD which actually executes
;  4592          4692  1 !       the command.
;  4593          4693  1 !
;  4594          4694  1 ! INPUTS
;  4595          4695  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  4596          4696  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  4597          4697  1 !                    first character after the SELECT keyword.
;  4598          4698  1 !
;  4599          4699  1 !       VERB_NODE - A pointer to the verb node to be built up for the command
;  4600          4700  1 !                   being parsed.
;  4601          4701  1 !
;  4602          4702  1 ! OUTPUTS
;  4603          4703  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  4604          4704  1 !                    first character after the end of the command.  This normally
;  4605          4705  1 !                    means that the input string is exhausted.
;  4606          4706  1 !
;  4607          4707  1 !       VERB_NODE - The passed-in verb node is filled in so that it and its
;  4608          4708  1 !                   attached noun node contain all information picked up during
;  4609          4709  1 !                   the parse of the SELECT command.
;  4610          4710  1 !
;  4611          4711  1
;  4612          4712  2     BEGIN
;  4613          4713  2
;  4614          4714  2     MAP
;  4615          4715  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  4616          4716  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input verb node
;  4617          4717  2
;  4618          4718  2     LOCAL
;  4619          4719  2         DISPID: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;  4620          4720  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to ASCIC name of display
;  4621          4721  2         NO_QUALIFIERS,                  ! Flag set if no qualifiers are present
;  4622          4722  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Pointer to Noun Node
;  4623          4723  2         SELECT_BITS: BITVECTOR[32];     ! The selection kind bits
;  4624          4724  2
;  4625          4725  2
;  4626          4726  2
;  4627          4727  2     ! Loop over all the qualifiers specified on the command.
;  4628          4728  2     !
;  4629          4729  2     SELECT_BITS = 0;
;  4630          4730  2     NO_QUALIFIERS = TRUE;
;  4631          4731  2     WHILE DBG$NMATCH(.INPUT_DESC, DBG$CS_SLASH, 1) DO
;  4632          4732  3         BEGIN
;  4633          4733  3         NO_QUALIFIERS = FALSE;
;  4634          4734  3
;  4635          4735  3
;  4636          4736  3         ! Now determine which specific qualifier we have and set the corre-
;  4637          4737  3         ! sponding selection kind bit.
;  4638          4738  3         !
;  4639          4739  3         SELECTONE TRUE OF
;  4640          4740  3             SET
;  4641          4741  3
```

```
;  4642          4742  3
;  4643          4743  3              ! Check for and process the /HISTORY qualifier.
;  4644          4744  3              !
;  4645          4745  3              [IF NOT .DBG$GL_DEVELOPER[7] THEN FALSE ELSE
;  4646          4746  3               DBG$NMATCH(.INPUT_DESC, DBG$CS_HISTORY, 1)]:
;  4647          4747  3                  SELECT_BITS[0] = TRUE;
;  4648          4748  3
;  4649          4749  3              ! Check for and process the /INPUT qualifier.
;  4650          4750  3              !
;  4651          4751  3              [IF NOT .DBG$GL_DEVELOPER[7] THEN FALSE ELSE
;  4652          4752  3               DBG$NMATCH(.INPUT_DESC, DBG$CS_INPUT, 1)]:
;  4653          4753  3                  SELECT_BITS[1] = TRUE;
;  4654          4754  3
;  4655          4755  3              ! Check for and process the /OUTPUT qualifier.
;  4656          4756  3              !
;  4657          4757  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_OUTPUT, 1)]:
;  4658          4758  3                  SELECT_BITS[2] = TRUE;
;  4659          4759  3
;  4660          4760  3              ! Check for and process the /SCROLLING qualifier.
;  4661          4761  3              !
;  4662          4762  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_SCROLLING, 2)]:
;  4663          4763  3                  SELECT_BITS[3] = TRUE;
;  4664          4764  3
;  4665          4765  3              ! Check for and process the /SOURCE qualifier.
;  4666          4766  3              !
;  4667          4767  3              [DBG$NMATCH(.INPUT_DESC, DBG$CS_SOURCE, 2)]:
;  4668          4768  3                  SELECT_BITS[4] = TRUE;
;  4669          4769  3
;  4670          4770  3              ! Any other qualifier constitutes a syntax error.
;  4671          4771  3              !
;  4672          4772  3              [OTHERWISE]:
;  4673          4773  3                  DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4674          4774  3
;  4675          4775  3              TES;
;  4676          4776  3
;  4677          4777  2          END;                            ! End of qualifier loop
;  4678          4778  2
;  4679          4779  2      ! If no qualifiers were specified, assume that /SCROLLING was meant.
;  4680          4780  2      !
;  4681          4781  2      IF .NO_QUALIFIERS THEN SELECT_BITS[3] = TRUE;
;  4682          4782  2
;  4683          4783  2      ! Pick up the name of the selected display and convert it to a Display ID.
;  4684          4784  2      ! If no display name is specified, leave the Display ID as zero.
;  4685          4785  2      !
;  4686          4786  2      DISPID = 0;
;  4687          4787  2      IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1)
;  4688          4788  2      THEN
;  4689          4789  3          BEGIN
;  4690          4790  3          NAMEPTR = PARSE_DISPLAY_NAME(.INPUT_DESC, TRUE);
;  4691          4791  3          DISPID = DBG$SCR_LOOKUP_DISPLAY(.NAMEPTR);
;  4692          4792  3
;  4693          4793  3          ! Check for and process the /SOURCE qualifier.
;  4694          4794  3          !
;  4695          4795  3          [DBG$NMATCH(.INPUT_DESC, DBG$CS_SOURCE, 2)]:
;  4696          4796  3              SELECT_BITS[4] = TRUE;
;  4697          4797  3
;  4698          4798  3          ! Any other qualifier constitutes a syntax error.
```

```
;  4699          4799  3          IF .DISPID EQL 0 THEN SIGNAL(DBG$_NOSUCHDISP, 1, .NAMEPTR);
;  4700          4800  3          IF .DISPID[DBG$V_DISP_REMOVE] THEN SIGNAL(DBG$_DISNOTSEL);
;  4701          4801  2          END;
;  4702          4802  2
;  4703          4803  2
;  4704          4804  2      ! Fill in the selection kind and the specified Display ID into the Noun
;  4705          4805  2      ! Node.  That completes the Verb and Noun Nodes, and we return.
;  4706          4806  2      !
;  4707          4807  2      NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE);
;  4708          4808  2      NOUN_NODE[DBG$L_NOUN_VALUE] = .SELECT_BITS;
;  4709          4809  2      NOUN_NODE[DBG$L_NOUN_VALUE2] = .DISPID;
;  4710          4810  2      VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;
;  4711          4811  2      RETURN;
;  4712          4812  2
;  4713          4813  1      END;


                          01FC 00000          .ENTRY  DBG$SCR_PARSE_SELECT_CMD, Save R2,R3,R4,R5,-  ; 4685
                                                      R6,R7,R8                                      ;
                58 00000000G   00   9E 00002          MOVAB   LIB$SIGNAL, R8                        ;
                57 00000000G   00   9E 00009          MOVAB   DBG$GL_DEVELOPER, R7                  ;
                56 00000000'   EF   9E 00010          MOVAB   DBG$CS_SLASH, R6                      ;
                55 00000000G   00   9E 00017          MOVAB   DBG$NMATCH, R5                        ;
                53             01   7D 0001E          MOVQ    #1, NO_QUALIFIERS                     ; 4730
                52       04    AC   D0 00021          MOVL    INPUT_DESC, R2                        ; 4731
                               01   DD 00025 1$:       PUSHL   #1                                    ;
                         0044  8F   BB 00027          PUSHR   #^M<R2,R6>                            ;
                65             03   FB 0002B          CALLS   #3, DBG$NMATCH                        ;
                03             50   E8 0002E          BLBS    R0, 2$                                ;
                             0081   31 00031          BRW     13$                                   ;
                               53   D4 00034 2$:       CLRL    NO_QUALIFIERS                         ; 4733
                               67   95 00036          TSTB    DBG$GL_DEVELOPER                      ; 4745
                               04   19 00038          BLSS    3$                                    ;
                               50   D4 0003A          CLRL    R0                                    ;
                               0A   11 0003C          BRB     4$                                    ;
                               01   DD 0003E 3$:       PUSHL   #1                                    ; 4746
                            93 A6   9F 00040          PUSHAB  DBG$CS_HISTORY                        ;
                               52   DD 00043          PUSHL   R2                                    ;
                65             03   FB 00045          CALLS   #3, DBG$NMATCH                        ;
                01             50   D1 00048 4$:       CMPL    R0, #1                                ; 4745
                               05   12 0004B          BNEQ    6$                                    ;
                54             01   88 0004D          BISB2   #1, SELECT_BITS                       ; 4747
                               D3   11 00050 5$:       BRB     1$                                    ;
                               67   95 00052 6$:       TSTB    DBG$GL_DEVELOPER                      ; 4752
                               04   19 00054          BLSS    7$                                    ;
                               50   D4 00056          CLRL    R0                                    ;
                               0A   11 00058          BRB     8$                                    ;
                               01   DD 0005A 7$:       PUSHL   #1                                    ; 4753
                            9B A6   9F 0005C          PUSHAB  DBG$CS_INPUT                          ;
                               52   DD 0005F          PUSHL   R2                                    ;
                65             03   FB 00061          CALLS   #3, DBG$NMATCH                        ;
                01             50   D1 00064 8$:       CMPL    R0, #1                                ; 4752
                               05   12 00067          BNEQ    9$                                    ;
                54             02   88 00069          BISB2   #2, SELECT_BITS                       ; 4754
```

```
                        B7  11 0006C          BRB     1$
                        01  DD 0006E 9$:      PUSHL   #1                                          ;4759
                    C9  A6  9F 00070          PUSHAB  DBG$CS_OUTPUT
                        52  DD 00073          PUSHL   R2
                    65      FB 00075          CALLS   #3, DBG$NMATCH
                    01      D1 00078          CMPL    R0, #1
                        05  12 0007B          BNEQ    10$
                    54      88 0007D          BISB2   #4, SELECT_BITS                             ;4760
                        A3  11 00080          BRB     1$
                        02  DD 00082 10$:     PUSHL   #2                                          ;4765
                    F1  A6  9F 00084          PUSHAB  DBG$CS_SCROLLING
                        52  DD 00087          PUSHL   R2
                    65      FB 00089          CALLS   #3, DBG$NMATCH
                    01      D1 0008C          CMPL    R0, #1
                        05  12 0008F          BNEQ    11$
                    54      88 00091          BISB2   #8, SELECT_BITS                             ;4766
                        8F  11 00094          BRB     1$
                        02  DD 00096 11$:     PUSHL   #2                                          ;4771
                    02  A6  9F 00098          PUSHAB  DBG$CS_SOURCE
                        52  DD 0009B          PUSHL   R2
                    65      FB 0009D          CALLS   #3, DBG$NMATCH
                    01      D1 000A0          CMPL    R0, #1
                        05  12 000A3          BNEQ    12$
                    54      10 88 000A5       BISB2   #16, SELECT_BITS                            ;4772
                        A6  11 000A8          BRB     5$
                        52  DD 000AA 12$:     PUSHL   R2                                          ;4778
000000000G          00      01 FB 000AC       CALLS   #1, DBG$SYNTAX_ERROR
                        9B  11 000B3          BRB     5$                                          ;4731
                    03      53 E9 000B5 13$:  BLBC    NO_QUALIFIERS, 14$                          ;4787
                    54      08 88 000B8       BISB2   #8, SELECT_BITS
                        53  D4 000BB 14$:     CLRL    DISPID                                      ;4793
                        01  DD 000BD          PUSHL   #1                                          ;4794
                  FF79  C6  9F 000BF          PUSHAB  DBG$CS_CR
                        52  DD 000C3          PUSHL   R2
                65      FB 000C5          CALLS   #3, DBG$NMATCH
                32      50 E8 000C8       BLBS    R0, 16$
                        01  DD 000CB          PUSHL   #1                                          ;4797
                        52  DD 000CD          PUSHL   R2
              0000V     CF  02 FB 000CF       CALLS   #2, PARSE_DISPLAY_NAME
                        52  50 DO 000D4       MOVL    R0, NAMEPTR
                        52  DD 000D7          PUSHL   NAMEPTR                                     ;4798
              EF35      CF  01 FB 000D9       CALLS   #1, DBG$SCR_LOOKUP_DISPLAY
                        53  50 DO 000DE       MOVL    R0, DISPID
                        0D  12 000E1          BNEQ    15$                                         ;4799
                        52  DD 000E3          PUSHL   NAMEPTR
                        01  DD 000E5          PUSHL   #1
                00028A62    8F DD 000E7       PUSHL   #166498
                68      03  FB 000ED         CALLS   #3, LIB$SIGNAL
                09          0A A3 E9 000F0 15$:  BLBC  10(DISPID), 16$                            ;4800
                00028A92    8F DD 000F4       PUSHL   #166546
                68      01  FB 000FA         CALLS   #1, LIB$SIGNAL
                        04  DD 000FD 16$:     PUSHL   #4                                          ;4807
000000000G              00  01 FB 000FF       CALLS   #1, DBG$GET_TEMPMEM
                        60  54 DO 00106       MOVL    SELECT_BITS, (NOUN_NODE)                    ;4808
              0C        A0  53 DO 00109       MOVL    DISPID, 12(NOUN_NODE)                       ;4809
                        51      08 AC DO 0010D  MOVL  VERB_NODE, R1                               ;4810
              08        A1  50 DO 00111       MOVL    NOUN_NODE, 8(R1)
```

```
                04 00115        RET                                     ; 4813

; Routine Size:  278 bytes,    Routine Base:  DBG$CODE + 2A19
```

```
;  4715          4814  1 GLOBAL ROUTINE DBG$SCR_PARSE_SETTERM_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  4716          4815  1 !
;  4717          4816  1 ! FUNCTION
;  4718          4817  1 !       This routine parses the SET TERMINAL command.  It accepts a command
;  4719          4818  1 !       line string descriptor as input and produces a Verb Node for the
;  4720          4819  1 !       command as output.  That Verb Node and its attached Noun Node later
;  4721          4820  1 !       serve as input to EXECUTE_SETTERM_CMD, which actually executes the
;  4722          4821  1 !       command.  The SET TERMINAL command accepts no parameters and only
;  4723          4822  1 !       one qualifier, the /WIDTH:n qualifier, which is mandatory at present.
;  4724          4823  1 !
;  4725          4824  1 ! INPUTS
;  4726          4825  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  4727          4826  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  4728          4827  1 !                    first character after the SET TERMINAL keywords.
;  4729          4828  1 !
;  4730          4829  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  4731          4830  1 !                   being parsed.
;  4732          4831  1 !
;  4733          4832  1 ! OUTPUTS
;  47 4          4833  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  47 5          4834  1 !                    first character after the end of the command.  This normally
;  473           4835  1 !                    means that the input string is exhausted.
;  473           4836  1 !
;  4738          4837  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it and its
;  4739          4838  1 !                   attached Noun Node contain all information picked up during
;  4740          4839  1 !                   the parse of the SET TERMINAL command.
;  4741          4840  1 !
;  4742          4841  1
;  4743          4842  2     BEGIN
;  4744          4843  2
;  4745          4844  2     MAP
;  4746          4845  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  4747          4846  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  4748          4847  2
;  4749          4848  2     LOCAL
;  4750          4849  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Pointer to Noun Node
;  4751          4850  2         WIDTH;                          ! The parsed terminal width
;  4752          4851  2
;  4753          4852  2
;  4754          4853  2
;  4755          4854  2     ! Check for and parse the mandatory /WIDTH:n qualifier.
;  4756          4855  2     !
;  4757          4856  2     IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_SLASH, 1)
;  4758          4857  2     THEN
;  4759          4858  2         DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4760          4859  2
;  4761          4860  2     IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_WIDTH, 1)
;  4762          4861  2     THEN
;  4763          4862  2         DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4764          4863  2
;  4765          4864  2     IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COLON, 1)
;  4766          4865  2     THEN
;  4767          4866  3         BEGIN
;  4768          4867  3         IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_EQUAL, 1)
;  4769          4868  3         THEN
;  4770          4869  3             DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4771          4870  3
```

```
;  4772          4871  2          END;
;  4773          4872  2
;  4774          4873  2
;  4775          4874  2      ! Pick up the specified terminal width and check it for validity.  (We
;  4776          4875  2      ! force it to be in the range 72 - 132.)  Then allocate a Noun Node, store
;  4777          4876  2      ! the new width in the Noun Node, and return.
;  4778          4877  2      !
;  4779          4878  2      DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, WIDTH);
;  4780          4879  2      WIDTH = MAX(72, MIN(132, .WIDTH));
;  4781          4880  2      NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE);
;  4782          4881  2      NOUN_NODE[DBG$L_NOUN_VALUE] = .WIDTH;
;  4783          4882  2      VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;
;  4784          4883  2      RETURN;
;  4785          4884  2
;  4786          4885  1      END;


                                  003C 00000          .ENTRY  DBG$SCR_PARSE_SETTERM_CMD, Save R2,R3,R4,R5 ; 4814
                55 00000000G   00  9E 00002          MOVAB   DBG$SYNTAX_ERROR, R5                        ;
                54 00000000'   EF  9E 00009          MOVAB   DBG$CS_SLASH, R4                            ;
                53 00000000G   00  9E 00010          MOVAB   DBG$NMATCH, R3                              ;
                5E                 04  C2 00017      SUBL2   #4, SP                                      ;
                                   01  DD 0001A      PUSHL   #1                                          ; 4856
                                   54  DD 0001C      PUSHL   R4                                          ;
                52             04  AC  D0 0001E      MOVL    INPUT_DESC, R2                              ;
                                   52  DD 00022      PUSHL   R2                                          ;
                63                 03  FB 00024      CALLS   #3, DBG$NMATCH                              ;
                05                 50  E8 00027      BLBS    R0, 1$                                      ;
                                   52  DD 0002A      PUSHL   R2                                          ; 4858
                65                 01  FB 0002C      CALLS   #1, DBG$SYNTAX_ERROR                        ;
                                   01  DD 0002F  1$:  PUSHL   #1                                          ; 4860
                             10    A4  9F 00031      PUSHAB  DBG$CS_WIDTH                                ;
                                   52  DD 00034      PUSHL   R2                                          ;
                63                 03  FB 00036      CALLS   #3, DBG$NMATCH                              ;
                05                 50  E8 00039      BLBS    R0, 2$                                      ;
                                   52  DD 0003C      PUSHL   R2                                          ; 4862
                65                 01  FB 0003E      CALLS   #1, DBG$SYNTAX_ERROR                        ;
                                   01  DD 00041  2$:  PUSHL   #1                                          ; 4864
                           FF75    C4  9F 00043      PUSHAB  DBG$CS_COLON                                ;
                                   52  DD 00047      PUSHL   R2                                          ;
                63                 03  FB 00049      CALLS   #3, DBG$NMATCH                              ;
                12                 50  E8 0004C      BLBS    R0, 3$                                      ;
                                   01  DD 0004F      PUSHL   #1                                          ; 4867
                             83    A4  9F 00051      PUSHAB  DBG$CS_EQUAL                                ;
                                   52  DD 00054      PUSHL   R2                                          ;
                63                 03  FB 00056      CALLS   #3, DBG$NMATCH                              ;
                05                 50  E8 00059      BLBS    R0, 3$                                      ;
                                   52  DD 0005C      PUSHL   R2                                          ; 4869
                65                 01  FB 0005E      CALLS   #1, DBG$SYNTAX_ERROR                        ;
                           4004    8F  BB 00061  3$:  PUSHR   #^M<R2,SP>                                 ; 4878
00000000G       00                 02  FB 00065      CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER               ;
                50                 6E  D0 0006C      MOVL    WIDTH, R0                                   ; 4879
00000084        8F                 50  D1 0006F      CMPL    R0, #132                                    ;
                                   04  15 00076      BLEQ    4$                                          ;
```

```
                50      84    8F  9A 00078          MOVZBL  #132, R0                        ;
          00000048  8F          50  D1 0007C 4$:     CMPL    R0, #72                         ;
                                04  18 00083         BGEQ    5$                              ;
                50      48    8F  9A 00085         MOVZBL  #72, R0                         ;
                6E            50  D0 00089 5$:     MOVL    R0, WIDTH                       ;
                                04  DD 0008C         PUSHL   #4                              ; 4880
          00000000G 00          01  FB 0008E         CALLS   #1, DBG$GET_TEMPMEM             ;
                60              6E  D0 00095         MOVL    WIDTH, (NOUN_NODE)              ; 4881
                51      08      AC  D0 00098         MOVL    VERB_NODE, R1                   ; 4882
                08      A1          50  D0 0009C     MOVL    NOUN_NODE, 8(R1)                ;
                                    04 000A0         RET                                     ; 4885

; Routine Size:  161 bytes,    Routine Base:  DBG$CODE + 2B2F
```

```
;  4788          4886  1 GLOBAL ROUTINE DBG$SCR_PARSE_SETWIND_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  4789          4887  1 !
;  4790          4888  1 ! FUNCTION
;  4791          4889  1 !       This routine parses the SET WINDOW command. It accepts a command line
;  4792          4890  1 !       string descriptor as input and produces a Verb Node for the parsed SET
;  4793          4891  1 !       WINDOW command as output.  That Verb Node and its attached Noun Node
;  4794          4892  1 !       later serve as input to DBG$SCR_EXECUTE_SETWIND_CMD which actually
;  4795          4893  1 !       executes the command.
;  4796          4894  1 !
;  4797          4895  1 ! INPUTS
;  4798          4896  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  4799          4897  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  4800          4898  1 !                    first character after the SET WINDOW keywords.
;  4801          4899  1 !
;  4802          4900  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  4803          4901  1 !                   being parsed.
;  4804          4902  1 !
;  4805          4903  1 ! OUTPUTS
;  4806          4904  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  4807          4905  1 !                    first character after the end of the command.  This normally
;  4808          4906  1 !                    means that the input string is exhausted.
;  4809          4907  1 !
;  4810          4908  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it and all
;  4811          4909  1 !                   its attached Noun Nodes contains all information picked up
;  4812          4910  1 !                   during the parse of the SET WINDOW command.
;  4813          4911  1 !
;  4814          4912  1
;  4815          4913  2     BEGIN
;  4816          4914  2
;  4817          4915  2     MAP
;  4818          4916  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  4819          4917  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  4820          4918  2
;  4821          4919  2     LOCAL
;  4822          4920  2         CBEG,                           ! Beginning column location of window
;  4823          4921  2         CLEN,                           ! Column length (width) of window
;  4824          4922  2         COL_INFO,                       ! Encoded column parameters
;  4825          4923  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to ASCIC window name
;  4826          4924  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! Pointer to current Noun Node
;  4827          4925  2         RBEG,                           ! Beginning row location of window
;  4828          4926  2         RLEN,                           ! Row length (height) of window
;  4829          4927  2         ROW_INFO;                       ! Encoded row parameters
;  4830          4928  2
;  4831          4929  2
;  4832          4930  2
;  4833          4931  2     ! Start by picking up the name of the screen window being defined.
;  4834          4932  2     !
;  4835          4933  2     NAMEPTR = PARSE_WINDOW_NAME(.INPUT_DESC);
;  4836          4934  2
;  4837          4935  2
;  4838          4936  2     ! Next scan past the AT keyword and the opening parenthesis for the
;  4839          4937  2     ! window parameters list.  After finding the open parenthesis, we
;  4840          4938  2     ! parse all the window parameter values.
;  4841          4939  2     !
;  4842          4940  2     IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_AT, 2)
;  4843          4941  2     THEN
;  4844          4942  2         DBG$SYNTAX_ERROR(.INPUT_DESC);
```

```
;  4845          4943  2
;  4846          4944  2      IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_LPAREN, 1)
;  4847          4945  2      THEN
;  4848          4946  2          DBG$SYNTAX_ERROR(.INPUT_DESC);
;  4849          4947  2
;  4850          4948  2      PARSE_WINDOW_PARAMETERS(.INPUT_DESC, RBEG, RLEN, CBEG, CLEN);
;  4851          4949  2
;  4852          4950  2
;  4853          4951  2      ! Allocate a Noun Node, attach it to the Verb Node, and fill all the
;  4854          4952  2      ! window parameters into the Noun Node.  Then return.
;  4855          4953  2      !
;  4856          4954  2      NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE_LONG);
;  4857          4955  2      VERB_NODE[DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;
;  4858          4956  2      NOUN_NODE[DBG$L_NOUN_VALUE] = .NAMEPTR;
;  4859          4957  2      ROW_INFO<W0_> = .RBEG;
;  4860          4958  2      ROW_INFO<W1_> = .RLEN;
;  4861          4959  2      NOUN_NODE[DBG$L_NOUN_VALUE2] = .ROW_INFO;
;  4862          4960  2      COL_INFO<W0_> = .CBEG;
;  4863          4961  2      COL_INFO<W1_> = .CLEN;
;  4864          4962  2      NOUN_NODE[DBG$L_NOUN_VALUE3] = .COL_INFO;
;  4865          4963  2      RETURN;
;  4866          4964  2
;  4867          4965  1      END;

                        003C 00000         .ENTRY  DBG$SCR_PARSE_SETWIND_CMD, Save R2,R3,R4,R5 ; 4886
          55 00000000G  00   9E 00002      MOVAB   DBG$SYNTAX_ERROR, R5
          54 00000000G  00   9E 00009      MOVAB   DBG$NMATCH, R4
          5E            10   C2 00010      SUBL2   #16, SP
          52        04  AC   D0 00013      MOVL    INPUT_DESC, R2                          ; 4933
                        52   DD 00017      PUSHL   R2
0000V     CF            01   FB 00019      CALLS   #1, PARSE_WINDOW_NAME
          53            50   D0 0001E      MOVL    R0, NAMEPTR
                        02   DD 00021      PUSHL   #2                                      ; 4940
             00000000'  EF   9F 00023      PUSHAB  DBG$CS_AT
                        52   DD 00029      PUSHL   R2
          64            03   FB 0002B      CALLS   #3, DBG$NMATCH
          05            50   E8 0002E      BLBS    R0, 1$
                        52   DD 00031      PUSHL   R2                                      ; 4942
          65            01   FB 00033      CALLS   #1, DBG$SYNTAX_ERROR
                        01   DD 00036 1$:  PUSHL   #1                                      ; 4944
             00000000'  EF   9F 00038      PUSHAB  DBG$CS_LPAREN
                        52   DD 0003E      PUSHL   R2
          64            03   FB 00040      CALLS   #3, DBG$NMATCH
          05            50   E8 00043      BLBS    R0, 2$
                        52   DD 00046      PUSHL   R2                                      ; 4946
          65            01   FB 00048      CALLS   #1, DBG$SYNTAX_ERROR
                        5E   DD 0004B 2$:  PUSHL   SP                                      ; 4948
                    08  AE   9F 0004D      PUSHAB  CBEG
                    10  AE   9F 00050      PUSHAB  RLEN
                    18  AE   9F 00053      PUSHAB  RBEG
                        52   DD 00056      PUSHL   R2
0000V     CF            05   FB 00058      CALLS   #5, PARSE_WINDOW_PARAMETERS
                        06   DD 0005D      PUSHL   #6                                      ; 4954
```

```
                    00000000G  00          01  FB 0005F          CALLS   #1, DBG$GET_TEMPMEM                ;
                               51     08    AC  D0 00066          MOVL    VERB_NODE, R1                     ; 4955
                           08  A1           50  D0 0006A          MOVL    NOUN_NODE, 8(R1)                  ;
                               60           53  D0 0006E          MOVL    NAMEPTR, (NOUN_NODE)              ; 4956
                               51     0C    AE  B0 00071          MOVW    RBEG, ROW_INFO                    ; 4957
          51            10                 10  08  AE  F0 00075  INSV    RLEN, #16, #16, ROW_INFO          ; 4958
                           0C  A0           51  D0 0007B          MOVL    ROW_INFO, 12(NOUN_NODE)           ; 4959
                               51     04    AE  B0 0007F          MOVW    CBEG, COL_INFO                    ; 4960
          51            10                 10  6E  F0 00083  INSV    CLEN, #16, #16, COL_INFO          ; 4961
                           10  A0           51  D0 00088          MOVL    COL_INFO, 16(NOUN_NODE)           ; 4962
                                              04 0008C          RET                                       ; 4965

; Routine Size:  141 bytes,    Routine Base:  DBG$CODE + 2BD0
```

```
;  4869          4966  1 GLOBAL ROUTINE DBG$SCR_PARSE_SHODISP_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  4870          4967  1 !
;  4871          4968  1 ! FUNCTION
;  4872          4969  1 !       This routine parses the SHOW DISPLAY command.  It accepts a command
;  4873          4970  1 !       line string descriptor as input and produces a Verb Node for the parsed
;  4874          4971  1 !       SHOW DISPLAY command as output.  That Verb Node later serves as input
;  4875          4972  1 !       to DBG$SCR_EXECUTE_SHODISP_CMD which actually executes the command.
;  4876          4973  1 !
;  4877          4974  1 ! INPUTS
;  4878          4975  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  4879          4976  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  4880          4977  1 !                    first character after the SHOW DISPLAY keywords.
;  4881          4978  1 !
;  4882          4979  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  4883          4980  1 !                   being parsed.
;  4884          4981  1 !
;  4885          4982  1 ! OUTPUTS
;  4886          4983  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  4887          4984  1 !                    first character after the end of the command.  This normally
;  4888          4985  1 !                    means that the input string is exhausted.
;  4889          4986  1 !
;  4890          4987  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it contains
;  4891          4988  1 !                   all information picked up during the parse of the SHOW
;  4892          4989  1 !                   DISPLAY command.
;  4893          4990  1 !
;  4894          4991  1
;  4895          4992  2     BEGIN
;  4896          4993  2
;  4897          4994  2     MAP
;  4898          4995  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  4899          4996  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  4900          4997  2
;  4901          4998  2     LOCAL
;  4902          4999  2         XXXXXXX;                        !<<--------- Local declarations --------
;  4903          5000  2
;  4904          5001  2
;  4905          5002  2
;  4906          5003  2     ! At present the SHOW DISPLAY command has no qualifiers or parameters.
;  4907          5004  2     ! Hence there is nothing to parse and we return immediately.
;  4908          5005  2     !
;  4909          5006  2     RETURN;
;  4910          5007  2
;  4911          5008  1     END;


                                                0000 00000         .ENTRY  DBG$SCR_PARSE_SHODISP_CMD, Save nothing     ; 4966
                                                  04 00002         RET                                                 ; 5008

; Routine Size:  3 bytes,    Routine Base:  DBG$CODE + 2C5D
```

```
;  4913          5009  1 GLOBAL ROUTINE DBG$SCR_PARSE_SHOWIND_CMD(INPUT_DESC, VERB_NODE): NOVALUE =
;  4914          5010  1 !
;  4915          5011  1 ! FUNCTION
;  4916          5012  1 !       This routine parses the SHOW WINDOW command. It accepts a command line
;  4917          5013  1 !       string descriptor as input and produces a Verb Node for the parsed SHOW
;  4918          5014  1 !       WINDOW command as output.  That Verb Node later serves as input to
;  4919          5015  1 !       DBG$SCR_EXECUTE_SHOWIND_CMD which actually executes the command.
;  4920          5016  1 !
;  4921          5017  1 ! INPUTS
;  4922          5018  1 !       INPUT_DESC - A string descriptor pointing to the input line being
;  4923          5019  1 !                    parsed.  The descriptor is assumed to be pointing to the
;  4924          5020  1 !                    first character after the SHOW WINDOW keywords.
;  4925          5021  1 !
;  4926          5022  1 !       VERB_NODE - A pointer to the Verb Node to be built up for the command
;  4927          5023  1 !                   being parsed.
;  4928          5024  1 !
;  4929          5025  1 ! OUTPUTS
;  4930          5026  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  4931          5027  1 !                    first character after the end of the command.  This normally
;  4932          5028  1 !                    means that the input string is exhausted.
;  4933          5029  1 !
;  4934          5030  1 !       VERB_NODE - The passed-in Verb Node is filled in so that it contains
;  4935          5031  1 !                   all information picked up during the parse of the SHOW WINDOW
;  4936          5032  1 !                   command.
;  4937          5033  1 !
;  4938          5034  1
;  4939          5035  2     BEGIN
;  4940          5036  2
;  4941          5037  2     MAP
;  4942          5038  2         INPUT_DESC: REF BLOCK[,BYTE],   ! Pointer to input string descriptor
;  4943          5039  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to input Verb Node
;  4944          5040  2
;  4945          5041  2     LOCAL
;  4946          5042  2         XXXXXXX;                        !<<--------- Local declarations --------
;  4947          5043  2
;  4948          5044  2
;  4949          5045  2
;  4950          5046  2     ! At present the SHOW WINDOW command has no qualifiers or parameters.
;  4951          5047  2     ! Hence there is nothing to parse and we return immediately.
;  4952          5048  2     !
;  4953          5049  2     RETURN;
;  4954          5050  2
;  4955          5051  1     END;


                                        0000 00000          .ENTRY  DBG$SCR_PARSE_SHOWIND_CMD, Save nothing    ; 5009
                                          04 00002          RET                                                ; 5051

; Routine Size:  3 bytes,    Routine Base:  DBG$CODE + 2C60
```

```
;  4957          5052  1 GLOBAL ROUTINE DBG$SCR_READ_LINE(RABPTR) =
;  4958          5053  1 !
;  4959          5054  1 ! FUNCTION
;  4960          5055  1 !       This routine reads a line of command input from the user's terminal.
;  4961          5056  1 !       If a current input display has been selected, the input line is read
;  4962          5057  1 !       from that display; otherwise the line is read from the user program
;  4963          5058  1 !       scrolling window.  The routine in effect simulates a call to the $GET
;  4964          5059  1 !       system service: It accepts a pointer to a RAB as input and it produces
;  4965          5060  1 !       a status code as output.  All information about the the prompt string
;  4966          5061  1 !       to be used and the input buffer to be filled is gotten from the RAB
;  4967          5062  1 !       and all outputs other than the status code (for example, the actual
;  4968          5063  1 !       input length) are returned to the RAB.
;  4969          5064  1 !
;  4970          5065  1 !       This routine should not be called if input is coming from an indirect
;  4971          5066  1 !       command file--it handles terminal input only.
;  4972          5067  1 !
;  4973          5068  1 ! INPUTS
;  4974          5069  1 !       RABPTR  - A pointer to an RMS Record Access Block (RAB) for the
;  4975          5070  1 !                 desired input operation.  This RAB must contain the proper
;  4976          5071  1 !                 input buffer and prompt string information for the I/O
;  4977          5072  1 !                 operation to be performed.
;  4978          5073  1 !
;  4979          5074  1 ! OUTPUTS
;  4980          5075  1 !       RABPTR  - RABPTR itself is not modified, but the RAB it points to is
;  4981          5076  1 !                 changed to indicate the actual length of the input string
;  4982          5077  1 !                 that was read from the appropriate screen display.  Other
;  4983          5078  1 !                 fields that could be changed by a $GET operation may also
;  4984          5079  1 !                 by changed by this routine.
;  4985          5080  1 !
;  4986          5081  1 !       An RMS status code as could be produced by $GET is returned as this
;  4987          5082  1 !                 routine's value.  In particular, if the input operation is
;  4988          5083  1 !                 successful, SS$_NORMAL is returned.
;  4989          5084  1 !
;  4990          5085  1
;  4991          5086  2     BEGIN
;  4992          5087  2
;  4993          5088  2     MAP
;  4994          5089  2         RABPTR: REF BLOCK[,BYTE];       ! Pointer to input RAB
;  4995          5090  2
;  4996          5091  2     LOCAL
;  4997          5092  2         XXXXXXX;                        !<<--------- Local declarations --------
;  4998          5093  2
;  4999          5094  2
;  5000          5095  2
;  5001          5096  2     ! The text of the routine starts here.
;  5002          5097  2     !
;  5003          5098  2     !<<-------------------------- FIRST LINE OF CODE -------------------------
;  5004          5099  2     RETURN SS$_NORMAL;
;  5005          5100  2
;  5006          5101  1     END;


                                  0000 00000         .ENTRY  DBG$SCR_READ_LINE, Save nothing          ; 5052
                       50      01  D0 00002         MOVL    #1, R0                                   ; 5099
```

```
                04 00005          RET                                     ; 5101

; Routine Size:  6 bytes,    Routine Base:  DBG$CODE + 2C63
```

```
;  5008          5102  1 GLOBAL ROUTINE DBG$SCR_SCREEN_MODE(SET_FLAG): NOVALUE =
;  5009          5103  1 !
;  5010          5104  1 ! FUNCTION
;  5011          5105  1 !       This routine executes the SET MODE SCREEN and SET MODE NOSCREEN
;  5012          5106  1 !       commands to set or clear "screen mode".  When screen mode is set,
;  5013          5107  1 !       screen displays will be displayed on the terminal in the normal
;  5014          5108  1 !       manner.  However, when screen mode is cancelled, screen displays
;  5015          5109  1 !       are not shown on the terminal even though the user may have a
;  5016          5110  1 !       number of active displays defined.  Those displays can be re-
;  5017          5111  1 !       enabled by setting screen mode again.
;  5018          5112  1 !
;  5019          5113  1 !       Screen mode is set by setting the global flag DBG$GL_SCREEN_MODE
;  5020          5114  1 !       to TRUE.  The first time screen mode is set, the pre-define displays
;  5021          5115  1 !       SRC, REG, and OUT are also defined.  Terminal input and output are
;  5022          5116  1 !       directed to the appropriate screen displays as well.
;  5023          5117  1 !
;  5024          5118  1 !       Screen mode is cancelled by clearing DBG$GL_SCREEN_MODE.  In addi-
;  5025          5119  1 !       tion all screen input and output is cancelled so that subsequent
;  5026          5120  1 !       output goes through RMS in non-screen mode.
;  5027          5121  1 !
;  5028          5122  1 ! INPUTS
;  5029          5123  1 !       SET_FLAG - If this flag is TRUE, screen mode is set by this routine.
;  5030          5124  1 !                  If the flag is FALSE, screen mode is cancelled.
;  5031          5125  1 !
;  5032          5126  1 ! OUTPUTS
;  5033          5127  1 !       NONE
;  5034          5128  1 !
;  5035          5129  1
;  5036          5130  2     BEGIN
;  5037          5131  2
;  5038          5132  2     LOCAL
;  5039          5133  2         CMDPTR: REF VECTOR[,WORD],      ! Pointer to the SRC command buffer
;  5040          5134  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;  5041          5135  2         SCREEN_INFO: BLOCK[12,BYTE],    ! Screen information control block
;  5042          5136  2         STATUS,                         ! Screen package status code
;  5043          5137  2         WPTR: REF DBG$WINDOW_ENTRY;     ! Pointer to a Screen Window Entry
;  5044          5138  2
;  5045          5139  2
;  5046          5140  2
;  5047          5141  2     ! If SET_FLAG is TRUE, we are executing the SET MODE SCREEN command and we
;  5048          5142  2     ! set "screen mode".
;  5049          5143  2     !
;  5050          5144  2     IF .SET_FLAG
;  5051          5145  2     THEN
;  5052          5146  3         BEGIN
;  5053          5147  3         DBG$GL_SCREEN_MODE = TRUE;
;  5054          5148  3
;  5055          5149  3
;  5056          5150  3         ! The first time screen mode is set, we direct screen output to logical
;  5057          5151  3         ! name DBG$OUTPUT if this name is defined.  We also "predefine" the SRC
;  5058          5152  3         ! source display, the REG register display, and the OUT output display.
;  5059          5153  3         ! SRC is marked as removed if the current language is MACRO or BASIC and
;  5060          5154  3         ! REG is marked as removed if the current language is anything else.
;  5061          5155  3         !
;  5062          5156  3         IF .SCREEN_MODE_FIRST_TIME
;  5063          5157  3         THEN
;  5064          5158  4             BEGIN
```

```
;  5065          5159  4
;  5066          5160  4
;  5067          5161  4              ! If the logical name DBG$OUTPUT exists, redirect the screen output
;  5068          5162  4              ! to that logical name.  Otherwise it goes to SYS$OUTPUT.
;  5069          5163  4              !
;  5070          5164  4              REDIRECT_SCREEN_OUTPUT();
;  5071          5165  4
;  5072          5166  4
;  5073          5167  4              ! Now get some information on the screen terminal.  The only piece
;  5074          5168  4              ! of information we actually use is whether this is a VT100 class
;  5075          5169  4              ! terminal or not.  If it is not, we disable the use of scrolling
;  5076          5170  4              ! regions (VT52s do not support scrolling regions).
;  5077          5171  4              !
;  5078          5172  4              STATUS = SCR$SCREEN_INFO(SCREEN_INFO);
;  5079          5173  4              IF NOT .STATUS THEN SIGNAL(.STATUS);
;  5080          5174  4              IF NOT .SCREEN_INFO[0, V_(1)] THEN DBG$GL_VT100_FLAG = FALSE;
;  5081          5175  4
;  5082          5176  4
;  5083          5177  4              ! Create source display SRC, an automatically updated display
;  5084          5178  4              ! which displays a window around the currently executed source
;  5085          5179  4              ! line.  Select SRC to be the current source display and the
;  5086          5180  4              ! the current scrolling display unless the language is MACRO or
;  5087          5181  4              ! BASIC.  For languages MACRO and BASIC, which do not support
;  5088          5182  4              ! source display, we mark this display as being removed.
;  5089          5183  4              !
;  5090          5184  4              WPTR = DBG$SCR_CREATE_TEMP_WINDOW(1, 9, 1, 132);
;  5091          5185  4              CMDPTR = DBG$GET_MEMORY(247%UPVAL + 1);
;  5092          5186  4              CH$MOVE(21, UPLIT BYTE(%ASCII 'EXAMINE/SOURCE .0\%PC'), CMDPTR[1]);
;  5093          5187  4              CMDPTR[0] = 22;
;  5094          5188  4              DISPTR = DBG$SCR_CREATE_DISPLAY(UPLIT BYTE(%ASCIC 'SRC'),
;  5095          5189  4                                              DBG$K_DISP_SOURCE, .WPTR, 50, .CMDPTR);
;  5096          5190  4              IF (.DBG$GB_LANGUAGE EQL DBG$K_MACRO) OR
;  5097          5191  5                 (.DBG$GB_LANGUAGE EQL DBG$K_BASIC)
;  5098          5192  4              THEN
;  5099          5193  4                  DISPTR[DBG$V_DISP_REMOVE] = TRUE
;  5100          5194  4
;  5101          5195  4              ELSE
;  5102          5196  5                  BEGIN
;  5103          5197  5                  DBG$SCR_CURDISP_SOURCE = .DISPTR;
;  5104          5198  5                  DBG$SCR_CURDISP_SCROLL = .DISPTR;
;  5105          5199  4                  END;
;  5106          5200  4
;  5107          5201  4
;  5108          5202  4              ! Create register display REG.  This display is marked as removed
;  5109          5203  4              ! for all languages other than MACRO and BASIC.
;  5110          5204  4              !
;  5111          5205  4              WPTR = DBG$SCR_CREATE_TEMP_WINDOW(1, 5, 1, 132);
;  5112          5206  4              DISPTR = DBG$SCR_CREATE_DISPLAY(UPLIT BYTE(%ASCIC 'REG'),
;  5113          5207  4                                              DBG$K_DISP_REGISTER, .WPTR, 5, 0);
;  5114          5208  4              IF (.DBG$GB_LANGUAGE EQL DBG$K_MACRO) OR
;  5115          5209  5                 (.DBG$GB_LANGUAGE EQL DBG$K_BASIC)
;  5116          5210  4              THEN
;  5117          5211  4                  REGISTER_DISPLAY(.DISPTR, DBG$RUNFRAME[DBG$L_USER_REGS])
;  5118          5212  4
;  5119          5213  4              ELSE
;  5120          5214  5                  BEGIN
;  5121          5215  5                  DISPTR[DBG$V_DISP_REMOVE] = TRUE;
```

```
;  5122          5216  5                DISPTR[DBG$W_DISP_RBEG] = 7;
;  5123          5217  4                END;
;  5124          5218  4
;  5125          5219  4
;  5126          5220  4            ! Create display OUT, the standard output display.  Select it as
;  5127          5221  4            ! the current output display.  If the current language is MACRO or
;  5128          5222  4            ! BASIC, we also select it as the current scrolling display and
;  5129          5223  4            ! adjust the window to fit with the REG display.
;  5130          5224  4            !
;  5131          5225  4            WPTR = DBG$SCR_CREATE_TEMP_WINDOW(11, 9, 1, 132);
;  5132          5226  4            DISPTR = DBG$SCR_CREATE_DISPLAY(UPLIT BYTE(%ASCIC 'OUT'),
;  5133          5227  4                                            DBG$K_DISP_NORMAL, .WPTR, 100, 0);
;  5134          5228  4            DBG$SCR_CURDISP_OUTPUT = .DISPTR;
;  5135          5229  4            IF (.DBG$GB_LANGUAGE EQL DBG$K_MACRO) OR
;  5136          5230  5               (.DBG$GB_LANGUAGE EQL DBG$K_BASIC)
;  5137          5231  4            THEN
;  5138          5232  5                BEGIN
;  5139          5233  5                DBG$SCR_CURDISP_SCROLL = .DISPTR;
;  5140          5234  5                DISPTR[DBG$W_DISP_RBEG] = 7;
;  5141          5235  5                DISPTR[DBG$W_DISP_RLEN] = 13;
;  5142          5236  4                END;
;  5143          5237  4
;  5144          5238  4
;  5145          5239  4            ! Clear the "first-time" flag so that we do not do the above
;  5146          5240  4            ! initialization the next time screen mode is entered.
;  5147          5241  4            !
;  5148          5242  4            SCREEN_MODE_FIRST_TIME = FALSE;
;  5149          5243  3            END;
;  5150          5244  3
;  5151          5245  3
;  5152          5246  3            ! Mark all lines in the "old" screen image as invalid.  This ensures
;  5153          5247  3            ! that DEBUG won't make any assumptions about the previous screen
;  5154          5248  3            ! contents when first putting displays on the screen.
;  5155          5249  3            !
;  5156          5250  3            INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  5157          5251  3                OLD_VALID[.I] = FALSE;
;  5158          5252  3
;  5159          5253  3
;  5160          5254  3            ! Set up the active input and output displays.  Here we set them to be
;  5161          5255  3            ! the currently selected input and output displays.  We then return.
;  5162          5256  3            !
;  5163          5257  3            DBG$GL_SCREEN_INPUT = .DBG$SCR_CURDISP_INPUT;
;  5164          5258  3            DBG$GL_SCREEN_OUTPUT = .DBG$SCR_CURDISP_OUTPUT;
;  5165          5259  3            DBG$GL_SCREEN_SOURCE = .DBG$SCR_CURDISP_SOURCE;
;  5166          5260  3            RETURN;
;  5167          5261  2            END;
;  5168          5262  2
;  5169          5263  2
;  5170          5264  2        ! If we get here, SET_FLAG is FALSE, so we cancel screen mode and return.
;  5171          5265  2        !
;  5172          5266  2        DBG$GL_SCREEN_MODE = FALSE;
;  5173          5267  2        DBG$GL_SCREEN_ERROR = 0;
;  5174          5268  2        DBG$GL_SCREEN_INPUT = 0;
;  5175          5269  2        DBG$GL_SCREEN_OUTPUT = 0;
;  5176          5270  2        DBG$GL_SCREEN_SOURCE = 0;
;  5177          5271  2        STATUS = SCR$SET_SCROLL(1, 24);
;  5178          5272  2        IF NOT .STATUS THEN SIGNAL(.STATUS);
```

```
;  5179          5273  2     RETURN;
;  5180          5274  2
;  5181          5275  1     END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

20 45 43 52 55 4F 53 2F 45 4E 49 4D 41 58 45  0067F P.AHL:   .ASCII  \EXAMINE/SOURCE .0\<92>\%PC\                    ;
                          43 50 25 5C 30 2E   0068E                                                               ;
                                43 52 53 03   00694 P.AHM:   .ASCII  <3>\SRC\                                     ;
                                47 45 52 03   00698 P.AHN:   .ASCII  <3>\REG\                                     ;
                                54 55 4F 03   0069C P.AHO:   .ASCII  <3>\OUT\                                     ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                          0FFC 00000            .ENTRY  DBG$SCR_SCREEN_MODE, Save R2,R3,R4,R5,R6,-  ; 5102
                                                                        R7,R8,R9,R10,R11                            ;
                          5B 00000000'   EF  9E 00002           MOVAB   P.AHL, R11                                  ;
                          5A 00000000'   EF  9E 00009           MOVAB   DBG$GL_SCREEN_MODE, R10                     ;
                          59 00000000'   EF  9E 00010           MOVAB   SCREEN_MODE_FIRST_TIME, R9                  ;
                          5E             0C  C2 00017           SUBL2   #12, SP                                     ;
                          03       04    AC  E8 0001A           BLBS    SET_FLAG, 1$                                ; 5144
                                   0136  31  0001E              BRW     16$                                         ;
                          6A             01  D0 00021 1$:       MOVL    #1, DBG$GL_SCREEN_MODE                      ; 5147
                          03             69  E8 00024           BLBS    SCREEN_MODE_FIRST_TIME, 2$                  ; 5156
                                   010E  31  00027              BRW     13$                                         ;
                    0000V CF             00  FB 0002A 2$:       CALLS   #0, REDIRECT_SCREEN_OUTPUT                  ; 5164
                                         5E  DD 0002F           PUSHL   SP                                          ; 5172
                00000000G 00             01  FB 00031           CALLS   #1, SCR$SCREEN_INFO                         ;
                          58             50  D0 00038           MOVL    R0, STATUS                                  ;
                          09             58  E8 0003B           BLBS    STATUS, 3$                                  ; 5173
                                         58  DD 0003E           PUSHL   STATUS                                      ;
                00000000G 00             01  FB 00040           CALLS   #1, LIB$SIGNAL                              ;
                    04    6E             01  E0 00047 3$:       BBS     #1, SCREEN_INFO, 4$                         ; 5174
                                E8B4     C9  D4 0004B           CLRL    DBG$GL_VT100_FLAG                           ;
                          7E      84     8F  9A 0004F 4$:       MOVZBL  #132, -(SP)                                 ; 5184
                                         01  DD 00053           PUSHL   #1                                          ;
                                         09  DD 00055           PUSHL   #9                                          ;
                                         01  DD 00057           PUSHL   #1                                          ;
                      DE46  CF           04  FB 00059           CALLS   #4, DBG$SCR_CREATE_TEMP_WINDOW              ;
                          57             50  D0 0005E           MOVL    R0, WPTR                                    ;
                                         07  DD 00061           PUSHL   #7                                          ; 5185
                00000000G 00             01  FB 00063           CALLS   #1, DBG$GET_MEMORY                          ;
                          56             50  D0 0006A           MOVL    R0, CMDPTR                                  ;
                    A6    6B             15  28 0006D           MOVC3   #21, P.AHL, 2(CMDPTR)                       ; 5186
                          66             16  B0 00072           MOVW    #22, (CMDPTR)                               ; 5187
                                         56  DD 00075           PUSHL   CMDPTR                                      ; 5189
                                         32  DD 00077           PUSHL   #50                                         ; 5188
                                         57  DD 00079           PUSHL   WPTR                                        ; 5189
                                         03  DD 0007B           PUSHL   #3                                          ; 5188
                                   15    AB  9F 0007D           PUSHAB  P.AHM                                       ;
                      DD88  CF           05  FB 00080           CALLS   #5, DBG$SCR_CREATE_DISPLAY                  ;
                          52             50  D0 00085           MOVL    R0, DISPTR                                  ;
                          50 00000000G   00  9A 00088           MOVZBL  DBG$GB_LANGUAGE, R0                         ; 5190
```

```
                        05  13 0008F          BEQL    5$
                    04  50  91 00091          CMPB    R0, #4                                                          ; 5191
                        06  12 00094          BNEQ    6$
                0A  A2      01  88 00096 5$:  BISB2   #1, 10(DISPTR)                                                  ; 5193
                        0A  11 0009A          BRB     7$
            E8C4    C9      52  D0 0009C 6$:  MOVL    DISPTR, DBG$SCR_CURDISP_SOURCE                                  ; 5197
            E8C0    C9      52  D0 000A1      MOVL    DISPTR, DBG$SCR_CURDISP_SCROLL                                  ; 5198
                7E      84  8F  9A 000A6 7$:  MOVZBL  #132, -(SP)                                                     ; 5205
                            01  DD 000AA      PUSHL   #1
                            05  DD 000AC      PUSHL   #5
                            01  DD 000AE      PUSHL   #1
            DDEF    CF      04  FB 000B0      CALLS   #4, DBG$SCR_CREATE_TEMP_WINDOW
                57          50  D0 000B5      MOVL    R0, WPTR
                7E          05  7D 000B8      MOVQ    #5, -(SP)                                                       ; 5206
                            57  DD 000BB      PUSHL   WPTR                                                            ; 5207
                            04  DD 000BD      PUSHL   #4                                                              ; 5206
                        19  AB  9F 000BF      PUSHAB  P.AHN
            DD46    CF      05  FB 000C2      CALLS   #5, DBG$SCR_CREATE_DISPLAY
                52          50  D0 000C7      MOVL    R0, DISPTR
                50  00000000G  00  9A 000CA   MOVZBL  DBG$GB_LANGUAGE, R0                                              ; 5208
                            05  13 000D1      BEQL    8$
                04          50  91 000D3      CMPB    R0, #4                                                          ; 5209
                            0F  12 000D6      BNEQ    9$
                    00000000G  00  9F 000D8 8$:  PUSHAB  DBG$RUNFRAME+4                                               ; 5211
                            52  DD 000DE      PUSHL   DISPTR
            0000V   CF      02  FB 000E0      CALLS   #2, REGISTER_DISPLAY
                            08  11 000E5      BRB     10$
                0A  A2      01  88 000E7 9$:  BISB2   #1, 10(DISPTR)                                                  ; 5215
                10  A2      07  B0 000EB      MOVW    #7, 16(DISPTR)                                                  ; 5216
                7E      84  8F  9A 000EF 10$:  MOVZBL  #132, -(SP)                                                    ; 5225
                            01  DD 000F3      PUSHL   #1
                            09  DD 000F5      PUSHL   #9
                            0B  DD 000F7      PUSHL   #11
            DDA6    CF      04  FB 000F9      CALLS   #4, DBG$SCR_CREATE_TEMP_WINDOW
                57          50  D0 000FE      MOVL    R0, WPTR
                            7E  D4 00101      CLRL    -(SP)                                                           ; 5226
                7E      64  8F  9A 00103      MOVZBL  #100, -(SP)
                            57  DD 00107      PUSHL   WPTR                                                            ; 5227
                            01  DD 00109      PUSHL   #1                                                              ; 5226
                        1D  AB  9F 0010B      PUSHAB  P.AHO
            DCFA    CF      05  FB 0010E      CALLS   #5, DBG$SCR_CREATE_DISPLAY
                52          50  D0 00113      MOVL    R0, DISPTR
            E8BC    C9      52  D0 00116      MOVL    DISPTR, DBG$SCR_CURDISP_OUTPUT                                  ; 5228
                50  00000000G  00  9A 0011B   MOVZBL  DBG$GB_LANGUAGE, R0                                             ; 5229
                            05  13 00122      BEQL    11$
                04          50  91 00124      CMPB    R0, #4                                                          ; 5230
                            0D  12 00127      BNEQ    12$
            E8C0    C9      52  D0 00129 11$:  MOVL    DISPTR, DBG$SCR_CURDISP_SCROLL                                  ; 5233
                10  A2  000D0007  8F  D0 0012E   MOVL    #851975, 16(DISPTR)                                             ; 5234
                            69  D4 00136 12$:  CLRL    SCREEN_MODE_FIRST_TIME                                          ; 5242
                            50  D4 00138 13$:  CLRL    I                                                               ; 5250
        00  FEFC  C9          50  E5 0013A 14$:  BBCC    I, OLD_VALID, 15$                                               ; 5251
        F6          50      14  F3 00140 15$:  AOBLEQ  #20, I, 14$
                F8  AA      E8B8  C9  D0 00144   MOVL    DBG$SCR_CURDISP_INPUT, DBG$GL_SCREEN_INPUT                      ; 5257
                08  AA      E8BC  C9  D0 0014A   MOVL    DBG$SCR_CURDISP_OUTPUT, -                                       ; 5258
                                                          DBG$GL_SCREEN_OUTPUT
                0C  AA      E8C4  C9  D0 00150   MOVL    DBG$SCR_CURDISP_SOURCE, -                                       ; 5259
```

```
                                                        DBG$GL_SCREEN_SOURCE
                                 04 00156               RET                                          ;  5146
                              6A 04 00157 16$:          CLRL    DBG$GL_SCREEN_MODE                   ;  5266
                           F0 AA 04 00159               CLRL    DBG$GL_SCREEN_ERROR                  ;  5267
                           F8 AA 04 0015C               CLRL    DBG$GL_SCREEN_INPUT                  ;  5268
                           08 AA 7C 0015F               CLRQ    DBG$GL_SCREEN_OUTPUT                 ;  5269
                              18 DD 00162               PUSHL   #24                                  ;  5271
                              01 DD 00164               PUSHL   #1                                   ;
                00000000G     02 FB 00166               CALLS   #2, SCR$SET_SCROLL                   ;
                           58 50 DD 0016D               MOVL    R0, STATUS                           ;
                           09 58 E8 00170               BLBS    STATUS, 17$                          ;  5272
                              58 DD 00173               PUSHL   STATUS                               ;
                00000000G     01 FB 00175               CALLS   #1, LIB$SIGNAL                       ;
                                 04 0017C 17$:          RET                                          ;  5275

; Routine Size:  381 bytes,    Routine Base:  DBG$CODE + 2C69
```

```
;  5183          5276  1 GLOBAL ROUTINE DBG$SCR_SCREEN_NORMAL: NOVALUE =
;  5184          5277  1 !
;  5185          5278  1 ! FUNCTION
;  5186          5279  1 !       This routine sets the screen back to "normal" if we are in Screen
;  5187          5280  1 !       Mode and it invalidates the current screen contents.  It is called
;  5188          5281  1 !       when we are about to break out from DEBUG with a SPAWN or ATTACH
;  5189          5282  1 !       command.  Hence the new process entered via SPAWN or ATTACH sees
;  5190          5283  1 !       a normal terminal screen with a full-screen scrolling region.  How-
;  5191          5284  1 !       ever, since the current screen contents are invalidated, the whole
;  5192          5285  1 !       screen is automatically refreshed when we later reenter DEBUG from
;  5193          5286  1 !       the SPAWN or ATTACH command.
;  5194          5287  1 !
;  5195          5288  1 ! INPUTS
;  5196          5289  1 !       NONE
;  5197          5290  1 !
;  5198          5291  1 ! OUTPUTS
;  5199          5292  1 !       NONE
;  5200          5293  1 !
;  5201          5294  1
;  5202          5295  2     BEGIN
;  5203          5296  2
;  5204          5297  2     LOCAL
;  5205          5298  2         STATUS;                         ! Status code from screen package
;  5206          5299  2
;  5207          5300  2
;  5208          5301  2
;  5209          5302  2     ! If Screen Mode is not set, there is nothing to do and we just return.
;  5210          5303  2     !
;  5211          5304  2     IF NOT .DBG$GL_SCREEN_MODE THEN RETURN;
;  5212          5305  2
;  5213          5306  2
;  5214          5307  2     ! Screen Mode is set.  Invalidate the current screen contents and set the
;  5215          5308  2     ! scrolling region to be the whole screen.  Then return.
;  5216          5309  2     !
;  5217          5310  2     INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  5218          5311  2         OLD_VALID[.I] = FALSE;
;  5219          5312  2
;  5220          5313  2     STATUS = SCR$SET_SCROLL(1, 24);
;  5221          5314  2     IF NOT .STATUS THEN SIGNAL(.STATUS);
;  5222          5315  2     RETURN;
;  5223          5316  2
;  5224          5317  1     END;


                                  0000 00000         .ENTRY  DBG$SCR_SCREEN_NORMAL, Save nothing  ; 5276
                    25 00000000'  EF   E9 00002      BLBC    DBG$GL_SCREEN_MODE, 3$               ; 5304
                                  50   D4 00009      CLRL    I                                    ; 5310
00 00000000'        EF            50   E5 0000B 1$:  BBCC    I, OLD_VALID, 2$                     ; 5311
F4                  50            14   F3 00013 2$:  AOBLEQ  #20, I, 1$                          ;
                                  18   DD 00017      PUSHL   #24                                  ; 5313
                                  01   DD 00019      PUSHL   #1                                   ;
   00000000G        00            02   FB 0001B      CALLS   #2, SCR$SET_SCROLL                   ;
                    09            50   E8 00022      BLBS    STATUS, 3$                           ; 5314
                                  50   DD 00025      PUSHL   STATUS                               ;
```

```
                    00000000G  00          01  FB 00027          CALLS   #1, LIB$SIGNAL                          ;
                                               04 0002E 3$:       RET                                            ; 5317

; Routine Size:  47 bytes,    Routine Base:  DBG$CODE + 2DE6
```

```
;  5226          5318  1 GLOBAL ROUTINE DBG$SCR_SCREEN_TERM: NOVALUE =
;  5227          5319  1 !
;  5228          5320  1 ! FUNCTION
;  5229          5321  1 !       This routine refreshes the screen and re-enables screen output
;  5230          5322  1 !       redirection on behalf of the DEBUG termination handler (exit
;  5231          5323  1 !       handler) that is called when the user program terminates.  In
;  5232          5324  1 !       effect, the purpose of this routine is to undo the effects of
;  5233          5325  1 !       the Screen Package Exit Handler which turns off output redirec-
;  5234          5326  1 !       tion and resets the scrolling region to the whole screen.  Hence
;  5235          5327  1 !       this routine turns on output redirection if DBG$OUTPUT is defined
;  5236          5328  1 !       and it invalidates the screen so that the whole screen is re-
;  5237          5329  1 !       freshed the next time it is updated.
;  5238          5330  1 !
;  5239          5331  1 ! INPUTS
;  5240          5332  1 !       NONE
;  5241          5333  1 !
;  5242          5334  1 ! OUTPUTS
;  5243          5335  1 !       NONE
;  5244          5336  1 !
;  5245          5337  1
;  5246          5338  2     BEGIN
;  5247          5339  2
;  5248          5340  2
;  5249          5341  2
;  5250          5342  2     ! If Screen Mode is not set, there is nothing to do and we just return.
;  5251          5343  2     !
;  5252          5344  2     IF NOT .DBG$GL_SCREEN_MODE THEN RETURN;
;  5253          5345  2
;  5254          5346  2
;  5255          5347  2     ! Screen Mode is set.  Redirect screen output to DBG$OUTPUT if this
;  5256          5348  2     ! logical name is defined.  Then invalidate the current screen contents                                  ;
;  5257          5349  2     ! and set the scrolling region to be the whole screen.  Then return.
;  5258          5350  2     !
;  5259          5351  2     REDIRECT_SCREEN_OUTPUT();
;  5260          5352  2     DBG$SCR_SCREEN_NORMAL();
;  5261          5353  2     RETURN;
;  5262          5354  2
;  5263          5355  1     END;


                                        0000 00000          .ENTRY  DBG$SCR_SCREEN_TERM, Save nothing          ; 5318
                    09 00000000'    EF  E9 00002          BLBC    DBG$GL_SCREEN_MODE, 1$                     ; 5344
              0000V CF              00  FB 00009          CALLS   #0, REDIRECT_SCREEN_OUTPUT                 ; 5351
                BF  AF              00  FB 0000E          CALLS   #0, DBG$SCR_SCREEN_NORMAL                  ; 5352
                                        04 00012 1$:      RET                                                ; 5355

; Routine Size:  19 bytes,    Routine Base:  DBG$CODE + 2E15
```

```
;  5265          5356  1 ROUTINE DBG$SCR_SCREEN_TO_LOGFILE: NOVALUE =
;  5266          5357  1 !
;  5267          5358  1 ! FUNCTION
;  5268          5359  1 !       This routine logs the entire contents of the terminal screen to the
;  5269          5360  1 !       current log-file provided that screen mode is set and that screen
;  5270          5361  1 !       logging has been enabled.  It simply outputs all screen contents,
;  5271          5362  1 !       including the rendition information, to the log file in a readable
;  5272          5363  1 !       format.  Screen logging is useful for error reporting and is required
;  5273          5364  1 !       by the DEBUG Test System.
;  5274          5365  1 !
;  5275          5366  1 ! INPUTS
;  5276          5367  1 !       NONE
;  5277          5368  1 !
;  5278          5369  1 ! OUTPUTS
;  5279          5370  1 !       NONE
;  5280          5371  1 !
;  5281          5372  1
;  5282          5373  2     BEGIN
;  5283          5374  2
;  5284          5375  2     BIND
;  5285          5376  2         HEADER_LINE = UPLIT BYTE(%ASCII '!     +----+----1----+----2----+----3',
;  5286          5377  2                                 '----+----4----+----5----+----6----+----7----+----8+'),
;  5287          5378  2         HEADER_132 = UPLIT BYTE(%ASCII '!     +----+----9----+----0',
;  5288          5379  2                                 '----+----1----+----2----+----3--+'),
;  5289          5380  2         HEX_TABLE = UPLIT BYTE(%ASCII '0123456789ABCDEF'): VECTOR[,BYTE];
;  5290          5381  2
;  5291          5382  2     LOCAL
;  5292          5383  2         BUFFER: VECTOR[132,BYTE],       ! Output formatting buffer
;  5293          5384  2         FIRST_TIME_FLAG,                ! Flag set first time line is output
;  5294          5385  2         LENGTH;                         ! The length of the formatted log line
;  5295          5386  2
;  5296          5387  2
;  5297          5388  2
;  5298          5389  2     ! Output the header line for the first 80 columns of screen image.
;  5299          5390  2     !
;  5300          5391  2     DBG$WRITE_LOG_FILE(88, HEADER_LINE);
;  5301          5392  2
;  5302          5393  2
;  5303          5394  2     ! Output the first 80 columns of screen image to the log file.
;  5304          5395  2     !
;  5305          5396  2     INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  5306          5397  3         BEGIN
;  5307          5398  3
;  5308          5399  3
;  5309          5400  3         ! Format the log line for the current screen line and fill in the
;  5310          5401  3         ! text of the first 80 columns of screen image.
;  5311          5402  3         !
;  5312          5403  3         CH$FILL(' ', 112, BUFFER);
;  5313          5404  3         BUFFER[0] = '!';
;  5314          5405  3         BUFFER[2] = (((.I + 1)/10) MOD 10) + '0';
;  5315          5406  3         IF .BUFFER[2] EQL '0' THEN BUFFER[2] = ' ';
;  5316          5407  3         BUFFER[3] = ((.I + 1) MOD 10) + '0';
;  5317          5408  3         BUFFER[4] = ':';
;  5318          5409  3         BUFFER[6] = '!';
;  5319          5410  3         BUFFER[87] = '!';
;  5320          5411  3         LENGTH = 88;
;  5321          5412  3         CH$MOVE(MIN(80, .OLD_CNT[.I]), OLD_SCREEN[.I*132], BUFFER[7]);
```

```
;  5322          5413  3
;  5323          5414  3
;  5324          5415  3          ! Format the length of the screen line and fill it in at the end of
;  5325          5416  3          ! the log line.
;  5326          5417  3          !
;  5327          5418  3          CH$MOVE(8, UPLIT BYTE(%ASCII 'length ='), BUFFER[90]);
;  5328          5419  3          IF .OLD_CNT[.I] GTR 999 THEN BUFFER[98] = '*';
;  5329          5420  3          BUFFER[99] = ((.OLD_CNT[.I]/100) MOD 10) + '0';
;  5330          5421  3          BUFFER[100] = ((.OLD_CNT[.I]/10) MOD 10) + '0';
;  5331          5422  3          BUFFER[101] = (.OLD_CNT[.I] MOD 10) + '0';
;  5332          5423  3          LENGTH = 102;
;  5333          5424  3          IF .BUFFER[99] EQL '0' THEN BUFFER[99] = ' ';
;  5334          5425  4          IF (.BUFFER[99] EQL ' ') AND (.BUFFER[100] EQL '0')
;  5335          5426  3          THEN
;  5336          5427  3              BUFFER[100] = ' ';
;  5337          5428  3
;  5338          5429  3
;  5339          5430  3          ! If this line has a single rendition attribute for the whole line,
;  5340          5431  3          ! include that rendition code at the end of the line.
;  5341          5432  3          !
;  5342          5433  3          IF .OLD_REND[.I] NEQ 255
;  5343          5434  3          THEN
;  5344          5435  4              BEGIN
;  5345          5436  4              CH$MOVE(8, UPLIT BYTE(%ASCII ', rend ='), BUFFER[102]);
;  5346          5437  4              BUFFER[111] = .HEX_TABLE[.OLD_REND[.I] AND %X'0F'];
;  5347          5438  4              LENGTH = 112;
;  5348          5439  3              END;
;  5349          5440  3
;  5350          5441  3
;  5351          5442  3              ! Write the formatted line to the log file.  Then loop for more.
;  5352          5443  3              !
;  5353          5444  3              DBG$WRITE_LOG_FILE(.LENGTH, BUFFER);
;  5354          5445  2              END;
;  5355          5446  2
;  5356          5447  2
;  5357          5448  2          ! Output a trailing header line to the log file.
;  5358          5449  2          !
;  5359          5450  2          DBG$WRITE_LOG_FILE(88, HEADER_LINE);
;  5360          5451  2
;  5361          5452  2
;  5362          5453  2          ! Output columns 81 - 132 of screen image to the log file for those lines
;  5363          5454  2          ! on the screen that have text in that column range.
;  5364          5455  2          !
;  5365          5456  2          FIRST_TIME_FLAG = TRUE;
;  5366          5457  2          INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  5367          5458  3              BEGIN
;  5368          5459  3
;  5369          5460  3
;  5370          5461  3              ! Output columns 81 - 132 of this screen line only if text actually
;  5371          5462  3              ! exists in those columns.  Otherwise ignore the line.
;  5372          5463  3              !
;  5373          5464  3              IF .OLD_CNT[.I] GTR 80
;  5374          5465  3              THEN
;  5375          5466  4                  BEGIN
;  5376          5467  4
;  5377          5468  4
;  5378          5469  4                  ! If this is the first line of this kind, output the header line
```

```
;  5379          5470  4              ! just ahead of it.
;  5380          5471  4              !
;  5381          5472  4              IF .FIRST_TIME_FLAG
;  5382          5473  4              THEN
;  5383          5474  5                  BEGIN
;  5384          5475  5                  FIRST_TIME_FLAG = FALSE;
;  5385          5476  5                  DBG$WRITE_LOG_FILE(60, HEADER_132);
;  5386          5477  4                  END;
;  5387          5478  4
;  5388          5479  4
;  5389          5480  4              ! Format the line, fill in the text from columns 81 - 132, and
;  5390          5481  4              ! output it to the log file.
;  5391          5482  4              !
;  5392          5483  4              CH$FILL(' ', 60, BUFFER);
;  5393          5484  4              BUFFER[0] = '!';
;  5394          5485  4              BUFFER[2] = (((.I + 1)/10) MOD 10) + '0';
;  5395          5486  4              IF .BUFFER[2] EQL '0' THEN BUFFER[2] = ' ';
;  5396          5487  4              BUFFER[3] = ((.I + 1) MOD 10) + '0';
;  5397          5488  4              BUFFER[4] = ':';
;  5398          5489  4              BUFFER[6] = '!';
;  5399          5490  4              BUFFER[59] = '!';
;  5400          5491  4              CH$MOVE(MIN(52, .OLD_CNT[.I] - 80),
;  5401          5492  4                                  OLD_SCREEN[.I*132 + 80], BUFFER[7]);
;  5402          5493  4              DBG$WRITE_LOG_FILE(60, BUFFER);
;  5403          5494  3              END;
;  5404          5495  3
;  5405          5496  2          END;
;  5406          5497  2
;  5407          5498  2
;  5408          5499  2      ! If anything was output just above, output a trailing header line too.
;  5409          5500  2      !
;  5410          5501  2      IF NOT .FIRST_TIME_FLAG THEN DBG$WRITE_LOG_FILE(60, HEADER_132);
;  5411          5502  2
;  5412          5503  2
;  5413          5504  2      ! Output the rendition information for all lines of text that have per-
;  5414          5505  2      ! character rendition information.  The rendition information is repre-
;  5415          5506  2      ! sented as one hex digit of rendition bits per character position.
;  5416          5507  2      !
;  5417          5508  2      INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  5418          5509  3          BEGIN
;  5419          5510  3          IF .OLD_REND[.I] EQL 255
;  5420          5511  3          THEN
;  5421          5512  4              BEGIN
;  5422          5513  4
;  5423          5514  4
;  5424          5515  4              ! Format the line, fill in the rendition bits for columns 1 - 80,
;  5425          5516  4              ! and output the line to the log file.
;  5426          5517  4              !
;  5427          5518  4              CH$FILL(' ', 87, BUFFER);
;  5428          5519  4              BUFFER[0] = '!';
;  5429          5520  4              BUFFER[2] = (((.I + 1)/10) MOD 10) + '0';
;  5430          5521  4              IF .BUFFER[2] EQL '0' THEN BUFFER[2] = ' ';
;  5431          5522  4              BUFFER[3] = ((.I + 1) MOD 10) + '0';
;  5432          5523  4              BUFFER[4] = ':';
;  5433          5524  4              INCR J FROM 0 TO MIN(80, .OLD_CNT[.I]) - 1 DO
;  5434          5525  5                  BEGIN
;  5435          5526  5                  BUFFER[.J + 7] = .HEX_TABLE[
```

```
;  5436     5527  5                                .OLD_SCREEN_REND[.I*132 + .J] AND %X'0F'];
;  5437     5528  4                END;
;  5438     5529  4
;  5439     5530  4            DBG$WRITE_LOG_FILE(87, BUFFER);
;  5440     5531  4
;  5441     5532  4
;  5442     5533  4            ! If the line is longer than 80 columns, format the excess on a
;  5443     5534  4            ! separate line and output that line to the log file.
;  5444     5535  4            !
;  5445     5536  4            IF .OLD_CNT[.I] GTR 80
;  5446     5537  4            THEN
;  5447     5538  5                BEGIN
;  5448     5539  5                CH$FILL(' ', 59, BUFFER);
;  5449     5540  5                BUFFER[0] = '!';
;  5450     5541  5                BUFFER[4] = ':';
;  5451     5542  5                INCR J FROM 80 TO MIN(132, .OLD_CNT[.I]) - 1 DO
;  5452     5543  6                    BEGIN
;  5453     5544  6                    BUFFER[.J + 7 - 80] = .HEX_TABLE[
;  5454     5545  6                                .OLD_SCREEN_REND[.I*132 + .J] AND %X'0F'];
;  5455     5546  5                    END;
;  5456     5547  5
;  5457     5548  5                DBG$WRITE_LOG_FILE(59, BUFFER);
;  5458     5549  4                END;
;  5459     5550  4
;  5460     5551  3            END;
;  5461     5552  3
;  5462     5553  2        END;
;  5463     5554  2
;  5464     5555  2
;  5465     5556  2        ! The screen image has been logged.  Now return.
;  5466     5557  2        !
;  5467     5558  2        RETURN;
;  5468     5559  2
;  5469     5560  1        END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

2D 2D 2D 2B 2D 2D 2D 2D 2B 20 20 20 20 20 21  006A0 P.AHP:   .ASCII  \!     +----+----1----+----2----+----3\
2D 2D 2D 32 2D 2D 2D 2D 2B 2D 2D 2D 2D 31 2D  006AF
                        33 2D 2D 2D 2D 2B 2D  006BE
2B 2D 2D 2D 2D 34 2D 2D 2D 2D 2B 2D 2D 2D 2D  006C5           .ASCII  \----+----4----+----5----+----6----+----7\
36 2D 2D 2D 2D 2B 2D 2D 2D 2D 35 2D 2D 2D 2D  006D4
                  37 2D 2D 2D 2D 2B 2D 2D 2D  006E3
               2B 38 2D 2D 2D 2D 2B 2D 2D 2D  006ED           .ASCII  \----+----8+\
2D 2D 2D 2B 2D 2D 2D 2D 2B 20 20 20 20 20 21  006F8 P.AHQ:   .ASCII  \!     +----+----9----+----0\
                  30 2D 2D 2D 2D 2B 2D 2D 2D  00707
2B 2D 2D 2D 2D 31 2D 2D 2D 2D 2B 2D 2D 2D 2D  00713           .ASCII  \----+----1----+----2----+----3--+\
33 2D 2D 2D 2D 2B 2D 2D 2D 2D 32 2D 2D 2D 2D  00722
                                    2B 2D 2D  00731
45 44 43 42 41 39 38 37 36 35 34 33 32 31 30  00734 P.AHR:   .ASCII  \0123456789ABCDEF\
                                          46  00743
                     3D 20 68 74 67 6E 65 6C  00744 P.AHS:   .ASCII  \length =\
                     3D 20 64 6E 65 72 20 2C  0074C P.AHT:   .ASCII  \, rend =\

                                                HEADER_LINE=        P.AHP
```

```
                        HEADER_132=         P.AHQ
                        HEX_TABLE=          P.AHR

                                            .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                    0FFC 00000 DBG$SCR_SCREEN_TO_LOGFILE:
                                               .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 5356
                5B 00000000G   00    9E 00002        MOVAB   DBG$WRITE_LOG_FILE, R11
                5A 00000000'   EF    9E 00009        MOVAB   HEX_TABLE, R10
                59 00000000'   EF    9E 00010        MOVAB   OLD_CNT, R9
                5E       FF7C  CE    9E 00017        MOVAB   -132(SP), SP
                               FF6C  CA    9F 0001C  PUSHAB  HEADER_LINE
                7E         58  8F    9A 00020        MOVZBL  #88, -(SP)
                6B             02    FB 00024        CALLS   #2, DBG$WRITE_LOG_FILE
                               56    D4 00027        CLRL    I                                   ; 5396
    0070  8F         20        6E    00  2C 00029 1$: MOVC5   #0, (SP), #32, #112, BUFFER        ; 5403
                               6E          00030
                               6E    21    90 00031  MOVB    #33, BUFFER                         ; 5404
                         01    50    A6    9E 00034  MOVAB   1(R6), R0                           ; 5405
                   51          50    0A    C7 00038  DIVL3   #10, R0, R1
          7E       00          51    01    7A 0003C  EMUL    #1, R1, #0, -(SP)
          51       51          8E    0A    7B 00041  EDIV    #10, (SP)+, R1, R1
             02    AE          51    30    81 00046  ADDB3   #48, R1, BUFFER+2
                         02    AE    91    0004B     CMPB    BUFFER+2, #48                       ; 5406
                               04    12    0004F     BNEQ    2$
                02     AE             20    90 00051  MOVB    #32, BUFFER+2
          7E       00          50    01    7A 00055 2$: EMUL    #1, R0, #0, -(SP)               ; 5407
          50       50          8E    0A    7B 0005A  EDIV    #10, (SP)+, R0, R0
             03    AE          50    30    81 0005F  ADDB3   #48, R0, BUFFER+3
                04     AE             3A    90 00064  MOVB    #58, BUFFER+4                       ; 5408
                06     AE       7C    8F    90 00068  MOVB    #124, BUFFER+6                      ; 5409
                57     AE       7C    8F    90 0006D  MOVB    #124, BUFFER+87                     ; 5410
                               58    58    8F  9A 00072  MOVZBL  #88, LENGTH                     ; 5411
                               57        6946  D0 00076  MOVL    OLD_CNT[I], R7                  ; 5412
                               50          57  D0 0007A  MOVL    R7, R0
                   00000050    8F          50  D1 0007D  CMPL    R0, #80
                                           04  15 00084  BLEQ    3$
                               50    50    8F  9A 00086  MOVZBL  #80, R0
                   51                  56 00000084  8F  C5 0008A 3$: MULL3   #132, I, R1
             07    AE       6C A941              50  28 00092  MOVC3   R0, OLD_SCREEN[R1], BUFFER+7
             5A    AE       10   AA              08  28 00099  MOVC3   #8, P.AHS, BUFFER+90       ; 5418
                   000003E7    8F              57  D1 0009F  CMPL    R7, #999                    ; 5419
                                               04  15 000A6  BLEQ    4$
                   62     AE                   2A  90 000A8  MOVB    #42, BUFFER+98
                   50                  57 00000064  8F  C7 000AC 4$: DIVL3   #100, R7, R0       ; 5420
          7E       00                  50          01  7A 000B4  EMUL    #1, R0, #0, -(SP)
          50       50                  8E          0A  7B 000B9  EDIV    #10, (SP)+, R0, R0
             63    AE                  50          30  81 000BE  ADDB3   #48, R0, BUFFER+99
                   50                  57          0A  C7 000C3  DIVL3   #10, R7, R0           ; 5421
          7E       00                  50          01  7A 000C7  EMUL    #1, R0, #0, -(SP)
          50       50                  8E          0A  7B 000CC  EDIV    #10, (SP)+, R0, R0
             64    AE                  50          30  81 000D1  ADDB3   #48, R0, BUFFER+100
          7E       00                  57          01  7A 000D6  EMUL    #1, R7, #0, -(SP)       ; 5422
          50       50                  8E          0A  7B 000DB  EDIV    #10, (SP)+, R0, R0
             65    AE                  50          30  81 000E0  ADDB3   #48, R0, BUFFER+101
                               58          66  8F  9A 000E5  MOVZBL  #102, LENGTH                ; 5423
```

```
                                30      63    AE   91 000E9          CMPB    BUFFER+99, #48                          ; 5424
                                        04    12   000ED          BNEQ    5$
                             63 AE      20    90   000EF          MOVB    #32, BUFFER+99
                                30      63    AE   91 000F3 5$:    CMPB    BUFFER+99, #32                          ; 5425
                                        0A    12   000F7          BNEQ    6$
                                30      64    AE   91 000F9          CMPB    BUFFER+100, #48
                                        04    12   000FD          BNEQ    6$
                             64 AE      20    90   000FF          MOVB    #32, BUFFER+100                         ; 5427
                             FF 8F      54 A946 91 00103 6$:    CMPB    OLD_REND[I], #255                       ; 5433
                                        16    13   00109          BEQL    7$
                          66    AE      18    AA   00108          MOVC3   #8, P.AHT, BUFFER+102                   ; 5436
                       50 54 A946       04    EF   00111          EXTZV   #0, #4, OLD_REND[I], R0                 ; 5437
                             6F AE            6A40 90 00118          MOVB    HEX_TABLE[R0], BUFFER+111
                                58   70    8F   9A 0011D          MOVZBL  #112, LENGTH                            ; 5438
                                     4100  8F   BB 00121 7$:    PUSHR   #^M<R8,SP>                              ; 5444
                                6B          02   FB 00125          CALLS   #2, DBG$WRITE_LOG_FILE
                          FEFB       56    01   14   F1 00128          ACBL    #20, #1, I, 1$                          ; 5396
                                     FF6C CA   9F   0012E          PUSHAB  HEADER_LINE                             ; 5450
                                7E   58   8F   9A   00132          MOVZBL  #88, -(SP)
                                6B          02   FB 00136          CALLS   #2, DBG$WRITE_LOG_FILE
                                57          01   D0 00139          MOVL    #1, FIRST_TIME_FLAG                     ; 5456
                                            56   D4 0013C          CLRL    I                                       ; 5457
                                00000050 8F         6946 D1 0013E 8$:    CMPL    OLD_CNT[I], #80                         ; 5464
                                            7C   15   00146          BLEQ    12$
                                0A          57   E9   00148          BLBC    FIRST_TIME_FLAG, 9$                     ; 5472
                                            57   D4   0014B          CLRL    FIRST_TIME_FLAG                         ; 5475
                                     C4     AA   9F   0014D          PUSHAB  HEADER_132                              ; 5476
                                            3C   DD   00150          PUSHL   #60
                                6B          02   FB   00152          CALLS   #2, DBG$WRITE_LOG_FILE
                          3C          20    6E   00   2C 00155 9$:    MOVC5   #0, (SP), #32, #60, BUFFER              ; 5483
                                            6E        0015A
                                6E          21   90   0015B          MOVB    #33, BUFFER                             ; 5484
                                51          01   A6   9E 0015E          MOVAB   1(R6), R1                               ; 5485
                                50          51   0A   C7 00162          DIVL3   #10, R1, R0
                             7E 00          50   01   7A 00166          EMUL    #1, R0, #0, -(SP)
                             50 50          8E   0A   7B 0016B          EDIV    #10, (SP)+, R0, R0
                          02 AE             50   30   81 00170          ADDB3   #48, R0, BUFFER+2
                                30      02    AE   91 00175          CMPB    BUFFER+2, #48                           ; 5486
                                        04    12   00179          BNEQ    10$
                             02 AE      20    90   0017B          MOVB    #32, BUFFER+2
                             7E 00      51    01   7A 0017F 10$:    EMUL    #1, R1, #0, -(SP)                       ; 5487
                             50 50      8E    0A   7B 00184          EDIV    #10, (SP)+, R0, R0
                          03 AE         50    30   81 00189          ADDB3   #48, R0, BUFFER+3
                             04 AE      3A    90   0018E          MOVB    #58, BUFFER+4                           ; 5488
                             06 AE   7C 8F    90   00192          MOVB    #124, BUFFER+6                          ; 5489
                             3B AE   7C 8F    90   00197          MOVB    #124, BUFFER+59                         ; 5490
                          51         6946 00000050 8F   C3 0019C          SUBL3   #80, OLD_CNT[I], R1                     ; 5491
                                     34   51   D1   001A5          CMPL    R1, #52
                                          03   15   001A8          BLEQ    11$
                                     51   34   D0   001AA          MOVL    #52, R1
                          50         56 00000084 8F   C5 001AD 11$:    MULL3   #132, I, R0                             ; 5492
                          07 AE   00BC C940      51   28 001B5          MOVC3   R1, OLD_SCREEN+80[R0], BUFFER+7
                                     5E   DD   001BD          PUSHL   SP                                      ; 5493
                                     3C   DD   001BF          PUSHL   #60
                                6B          02   FB   001C1          CALLS   #2, DBG$WRITE_LOG_FILE
                       FF74       56    01   14   F1 001C4 12$:    ACBL    #20, #1, I, 8$                          ; 5457
                                08          57   E8   001CA          BLBS    FIRST_TIME_FLAG, 13$                    ; 5501
```

```
                                                C4    AA   9F 001CD          PUSHAB  HEADER_132                                      ;
                                                      3C   DD 001D0          PUSHL   #60                                             ;
                                                68    02   FB 001D2          CALLS   #2, DBG$WRITE_LOG_FILE                          ;
                                                      56   D4 001D5 13$:     CLRL    I                                               ; 5508
                                                FF    8F   54 A946 91 001D7 14$:     CMPB    OLD_REND[I], #255                       ; 5510
                                                           03 13 001DD          BEQL    15$
                                                         00C7 31 001DF          BRW     23$
0057    8F              20                      6E         00 2C 001E2 15$:     MOVC5   #0, (SP), #32, #87, BUFFER                      ; 5518
                                                           6E    001E9
                                                6E         21 90 001EA          MOVB    #33, BUFFER                                     ; 5519
                                                50    01   A6 9E 001ED          MOVAB   1(R6), R0                                       ; 5520
                        57                      50         0A C7 001F1          DIVL3   #10, R0, R7
              7E        00                      57         01 7A 001F5          EMUL    #1, R7, #0, -(SP)
              57        57                      8E         0A 7B 001FA          EDIV    #10, (SP)+, R7, R7
                  02    AE                      57         30 81 001FF          ADDB3   #48, R7, BUFFER+2
                                                30    02   AE 91 00204          CMPB    BUFFER+2, #48                                   ; 5521
                                                           04 12 00208          BNEQ    16$
                                          02    AE                 20 90 0020A  MOVB    #32, BUFFER+2
              7E        00                      50                 01 7A 0020E 16$: EMUL    #1, R0, #0, -(SP)                       ; 5522
              50        50                      8E                 0A 7B 00213  EDIV    #10, (SP)+, R0, R0
                  03    AE                      50                 30 81 00218  ADDB3   #48, R0, BUFFER+3
                                          04    AE                 3A 90 0021D  MOVB    #58, BUFFER+4                                   ; 5523
                                                52               6946 D0 00221  MOVL    OLD_CNT[I], R2                                  ; 5524
                                    00000050    8F                 52 D1 00225  CMPL    R2, #80
                                                                   04 15 0022C  BLEQ    17$
                                                52         50      8F 9A 0022E  MOVZBL  #80, R2
                        57                      56 00000084      8F C5 00232 17$: MULL3   #132, I, R7                                     ; 5527
                                                51                 01 CE 0023A  MNEGL   #1, J
                                                                   12 11 0023D  BRB     19$
                        50                      57                 51 C1 0023F 18$: ADDL3   J, R7, R0
              50    0B40 C940                   04                 00 EF 00243  EXTZV   #0, #4, OLD_SCREEN_REND[R0], R0
                                          07 AE41                6A40 90 0024B  MOVB    HEX_TABLE[R0], BUFFER+7[J]                      ; 5526
                                EA                51               52 F2 00251 19$: AOBLSS  R2, J, 18$                              ; 5524
                                                                   5E DD 00255  PUSHL   SP                                              ; 5530
                                                7E         57      8F 9A 00257  MOVZBL  #87, -(SP)
                                                68                 02 FB 0025B  CALLS   #2, DBG$WRITE_LOG_FILE
                                    00000050    8F               6946 D1 0025E  CMPL    OLD_CNT[I], #80                                 ; 5536
                                                                   41 15 00266  BLEQ    23$
              3B                20                      6E         00 2C 00268  MOVC5   #0, (SP), #32, #59, BUFFER                      ; 5539
                                                           6E    0026D
                                                6E         21 90 0026E          MOVB    #33, BUFFER                                     ; 5540
                                          04    AE                 3A 90 00271  MOVB    #58, BUFFER+4                                   ; 5541
                                                52               6946 D0 00275  MOVL    OLD_CNT[I], R2                                  ; 5542
                                    00000084    8F                 52 D1 00279  CMPL    R2, #132
                                                                   04 15 00280  BLEQ    20$
                                                52         84      8F 9A 00282  MOVZBL  #132, R2
                                                51         4F      8F 9A 00286 20$: MOVZBL  #79, J
                                                                   12 11 0028A  BRB     22$
                        50                      57                 51 C1 0028C 21$: ADDL3   J, R7, R0                                   ; 5545
              50    0B40 C940                   04                 00 EF 00290  EXTZV   #0, #4, OLD_SCREEN_REND[R0], R0
                                          B7 AE41                6A40 90 00298  MOVB    HEX_TABLE[R0], BUFFER-73[J]                     ; 5544
                                EA                51               52 F2 0029E 22$: AOBLSS  R2, J, 21$                              ; 5542
                                                                   5E DD 002A2  PUSHL   SP                                              ; 5548
                                                                   3B DD 002A4  PUSHL   #59
                                                68                 02 FB 002A6  CALLS   #2, DBG$WRITE_LOG_FILE
              FF28              56                      01         14 F1 002A9 23$: ACBL    #20, #1, I, 14$                             ; 5508
                                                                      04 002AF  RET                                                     ; 5560
```

; Routine Size: 688 bytes,   Routine Base: DBG$CODE + 2E28

```
;  5471          5561  1 ROUTINE DBG$SCR_SCROLL_DISPLAY(DISPID, DIRECTION, AMOUNT): NOVALUE =
;  5472          5562  1 !
;  5473          5563  1 ! FUNCTION
;  5474          5564  1 !       This routine scrolls a specified display in a specified direction.
;  5475          5565  1 !       The scrolling is done by adjusting the window specification associ-
;  5476          5566  1 !       ated with this display and by also adjusting the pasteboard appro-
;  5477          5567  1 !       priately if this display is present on the pasteboard.
;  5478          5568  1 !
;  5479          5569  1 ! INPUTS
;  5480          5570  1 !       DISPID  - The Display ID of the display to be scrolled.  This is a
;  5481          5571  1 !                 pointer to the Screen Display Entry of that display.
;  5482          5572  1 !
;  5483          5573  1 !       DIRECTION - The direction in which the display is to be scrolled.
;  5484          5574  1 !                 The following values are accepted:
;  5485          5575  1 !
;  5486          5576  1 !                       DBG$K_SCROLL_UP    -- Scroll the window up over the
;  5487          5577  1 !                                             display text.
;  5488          5578  1 !                       DBG$K_SCROLL_DOWN  -- Scroll the window down over the
;  5489          5579  1 !                                             display text.
;  5490          5580  1 !                       DBG$K_SCROLL_LEFT  -- Scroll the window to the left
;  5491          5581  1 !                                             over the display text.
;  5492          5582  1 !                       DBG$K_SCROLL_RIGHT -- Scroll the window to the right
;  5493          5583  1 !                                             over the display text.
;  5494          5584  1 !
;  5495          5585  1 !       AMOUNT  - The amount to scroll the display.  This is either the number
;  5496          5586  1 !                 of lines to scroll it up or down or the number of columns to
;  5497          5587  1 !                 scroll it to the left or right.  This amount must always be
;  5498          5588  1 !                 zero or positive.
;  5499          5589  1 !
;  5500          5590  1 ! OUTPUTS
;  5501          5591  1 !       NONE.
;  5502          5592  1 !
;  5503          5593  1
;  5504          5594  2     BEGIN
;  5505          5595  2
;  5506          5596  2     MAP
;  5507          5597  2         DISPID: REF DBG$DISP_ENTRY;      ! Pointer to Screen Display Entry for
;  5508          5598  2                                          !      display to be scrolled
;  5509          5599  2
;  5510          5600  2     LOCAL
;  5511          5601  2         DELTA,                           ! Actual scroll movement up or down
;  5512          5602  2         NEW_DROW,                        ! New DROW value when scolling up or down
;  5513          5603  2         WSPTR: REF DBG$DLINE_ENTRY;      ! Pointer to first line in display window
;  5514          5604  2
;  5515          5605  2
;  5516          5606  2
;  5517          5607  2     ! Check that the scrolling amount is zero or positive.  If it is not,
;  5518          5608  2     ! signal an internal DEBUG error.
;  5519          5609  2     !
;  5520          5610  2     IF .AMOUNT LSS 0 THEN $DBG_ERROR('DBGSCREEN\SCROLL_DISPLAY 10');
;  5521          5611  2
;  5522          5612  2
;  5523          5613  2     ! Case on the scrolling direction to select the proper window adjustment.
;  5524          5614  2     !
;  5525          5615  2     CASE .DIRECTION FROM DBG$K_SCROLL_UP TO DBG$K_SCROLL_RIGHT OF
;  5526          5616  2         SET
;  5527          5617  2
```

```
;  5528          5618  2
;  5529          5619  2          ! Scroll up the display.  Compute the new DROW value (the new row
;  5530          5620  2          ! location of the display window in the display text) and then compute
;  5531          5621  2          ! the difference between the old and new DROW values.  Then shift the
;  5532          5622  2          ! window pointer up the display text that number of lines and set the
;  5533          5623  2          ! scrolling count.
;  5534          5624  2          !
;  5535          5625  2          [DBG$K_SCROLL_UP]:
;  5536          5626  3              BEGIN
;  5537          5627  3              IF .DISPID[DBG$B_DISP_KIND] EQL DBG$K_DISP_SOURCE
;  5538          5628  3              THEN
;  5539          5629  3                  IF DBG$SCR_SCROLL_SOURCE_UP(.DISPID, .AMOUNT) THEN RETURN;
;  5540          5630  3
;  5541          5631  3              NEW_DROW = MAX(1, MIN(
;  5542          5632  3                  .DISPID[DBG$W_DISP_LINECNT] - .DISPID[DBG$W_DISP_RLEN] + 1,
;  5543          5633  3                  .DISPID[DBG$W_DISP_DROW] - .AMOUNT));
;  5544          5634  3              DELTA = .DISPID[DBG$W_DISP_DROW] - .NEW_DROW;
;  5545          5635  3              DISPID[DBG$W_DISP_DROW] = .NEW_DROW;
;  5546          5636  3              IF .DELTA GTR 0
;  5547          5637  3              THEN
;  5548          5638  4                  BEGIN
;  5549          5639  4                  DISPID[DBG$W_DISP_SCROLL] = .DISPID[DBG$W_DISP_SCROLL] - .DELTA;
;  5550          5640  4                  WSPTR = .DISPID[DBG$L_DISP_WINDOW_PTR];
;  5551          5641  4                  INCR I FROM 1 TO .DELTA DO
;  5552          5642  4                      WSPTR = .WSPTR[DBG$L_DLINE_BLINK];
;  5553          5643  4
;  5554          5644  4                  DISPID[DBG$L_DISP_WINDOW_PTR] = .WSPTR;
;  5555          5645  3                  END;
;  5556          5646  3
;  5557          5647  2              END;
;  5558          5648  2
;  5559          5649  2
;  5560          5650  2          ! Scroll down the display.  Compute the new DROW value (the new row
;  5561          5651  2          ! location of the display window in the display text) and then compute
;  5562          5652  2          ! the difference between the old and new DROW values.  Then shift the
;  5563          5653  2          ! window pointer down the display text that number of lines and set the
;  5564          5654  2          ! scrolling count.
;  5565          5655  2          !
;  5566          5656  2          [DBG$K_SCROLL_DOWN]:
;  5567          5657  3              BEGIN
;  5568          5658  3              IF .DISPID[DBG$B_DISP_KIND] EQL DBG$K_DISP_SOURCE
;  5569          5659  3              THEN
;  5570          5660  3                  IF DBG$SCR_SCROLL_SOURCE_DOWN(.DISPID, .AMOUNT) THEN RETURN;
;  5571          5661  3
;  5572          5662  3              NEW_DROW = MAX(1, MIN(
;  5573          5663  3                  .DISPID[DBG$W_DISP_LINECNT] - .DISPID[DBG$W_DISP_RLEN] + 1,
;  5574          5664  3                  .DISPID[DBG$W_DISP_DROW] + .AMOUNT));
;  5575          5665  3              DELTA = .NEW_DROW - .DISPID[DBG$W_DISP_DROW];
;  5576          5666  3              DISPID[DBG$W_DISP_DROW] = .NEW_DROW;
;  5577          5667  3              IF .DELTA GTR 0
;  5578          5668  3              THEN
;  5579          5669  4                  BEGIN
;  5580          5670  4                  DISPID[DBG$W_DISP_SCROLL] = .DISPID[DBG$W_DISP_SCROLL] + .DELTA;
;  5581          5671  4                  WSPTR = .DISPID[DBG$L_DISP_WINDOW_PTR];
;  5582          5672  4                  INCR I FROM 1 TO .DELTA DO
;  5583          5673  4                      WSPTR = .WSPTR[DBG$L_DLINE_FLINK];
;  5584          5674  4
```

```
;  5585     5675  4                DISPID[DBG$L_DISP_WINDOW_PTR] = .WSPTR;
;  5586     5676  3                END;
;  5587     5677  3
;  5588     5678  2            END;
;  5589     5679  2
;  5590     5680  2
;  5591     5681  2        ! Scroll to the left side of the display.  This only requires adjusting
;  5592     5682  2        ! the display column value (DCOL).  Note, however, that we do not allow
;  5593     5683  2        ! DCOL to become less than 1.
;  5594     5684  2        !
;  5595     5685  2        [DBG$K_SCROLL_LEFT]:
;  5596     5686  2            DISPID[DBG$W_DISP_DCOL] = MAX(1, .DISPID[DBG$W_DISP_DCOL] - .AMOUNT);
;  5597     5687  2
;  5598     5688  2
;  5599     5689  2        ! Scroll to the right side of the display.  This also only requires
;  5600     5690  2        ! adjusting the display column value (DCOL).  Note that we do not allow
;  5601     5691  2        ! DCOL to exceed 133.
;  5602     5692  2        !
;  5603     5693  2        [DBG$K_SCROLL_RIGHT]:
;  5604     5694  2            DISPID[DBG$W_DISP_DCOL] = MIN(133, .DISPID[DBG$W_DISP_DCOL] + .AMOUNT);
;  5605     5695  2
;  5606     5696  2
;  5607     5697  2        ! Any other scrolling direction constitutes an internal DEBUG error.
;  5608     5698  2        !
;  5609     5699  2        [INRANGE, OUTRANGE]:
;  5610     5700  2            $DBG_ERROR('DBGSCREEN\SCROLL_DISPLAY 20');
;  5611     5701  2
;  5612     5702  2        TES;
;  5613     5703  2
;  5614     5704  2
;  5615     5705  2    ! The scrolling is completed.  Now return.
;  5616     5706  2    !
;  5617     5707  2    RETURN;
;  5618     5708  2
;  5619     5709  1    END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

4F 52 43 53 5C 4E 45 45 52 43 53 47 42 44 1B  00754 P.AHU:   .ASCII  <27>\DBGSCREEN\<92>\SCROLL_DISPLAY 10\      ;
   30 31 20 59 41 4C 50 53 49 44 5F 4C 4C  00763
4F 52 43 53 5C 4E 45 45 52 43 53 47 42 44 1B  00770 P.AHV:   .ASCII  <27>\DBGSCREEN\<92>\SCROLL_DISPLAY 20\      ;
   30 32 20 59 41 4C 50 53 49 44 5F 4C 4C  0077F


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                001C 00000 DBG$SCR_SCROLL_DISPLAY:
                                                            .WORD   Save R2,R3,R4                               ; 5561
                                    54 00000000G   00  9E 00002 MOVAB   LIB$SIGNAL, R4                          ;
                                    53       0C    AC  D0 00009 MOVL    AMOUNT, R3                             ; 5610
                                                   11  18 0000D BGEQ    1$                                     ;
                                       00000000'   EF  9F 0000F PUSHAB  P.AHU                                  ;
                                                   01  DD 00015 PUSHL   #1                                     ;
                                       00028362    8F  DD 00017 PUSHL   #164706                               ;
```

```
                                64              03 FB 0001D             CALLS   #3, LIB$SIGNAL
                              03 01        08   AC CF 00020 1$:         CASEL   DIRECTION, #1, #3                       ;5615
                    00EB    00D9    0078        001A    00025 2$:       .WORD   3$-2$,-
                                                                                9$-2$,-
                                                                                18$-2$,-
                                                                                19$-2$
                                00000000'       EF 9F 0002D             PUSHAB  P.AHV                                   ;5700
                                                01 DD 00033             PUSHL   #1
                                00028362        8F DD 00035             PUSHL   #164706
                                64              03 FB 0003B             CALLS   #3, LIB$SIGNAL
                                                04 0003E                RET
                                52      04   AC D0 0003F 3$:          MOVL    DISPID, R2                              ;5627
                                03      08   A2 91 00043             CMPB    8(R2), #3
                                             0A 12 00047             BNEQ    4$                                      ;5629
                                             0C BB 00049             PUSHR   #^M<R2,R3>
                          0000V CF           02 FB 0004B             CALLS   #2, DBG$SCR_SCROLL_SOURCE_UP
                                5B           50 E8 00050             BLBS    R0, 10$
                                51      1E   A2 3C 00053 4$:         MOVZWL  30(R2), R1                              ;5632
                                50      12   A2 3C 00057             MOVZWL  18(R2), R0
                                51           50 C2 0005B             SUBL2   R0, R1
                                50      01   A1 9E 0005E             MOVAB   1(R1), R0
                                51      18   A2 3C 00062             MOVZWL  24(R2), R1                              ;5633
                                51           53 C2 00066             SUBL2   R3, R1
                                51           50 D1 00069             CMPL    R0, R1
                                             03 15 0006C             BLEQ    5$
                                50           51 D0 0006E             MOVL    R1, R0
                                             50 D5 00071 5$:         TSTL    R0                                      ;5631
                                             03 14 00073             BGTR    6$
                                50           01 D0 00075             MOVL    #1, R0
                                51      18   A2 3C 00078 6$:         MOVZWL  24(R2), DELTA                           ;5634
                                51           50 C2 0007C             SUBL2   NEW_DROW, DELTA
                          18    A2           50 B0 0007F             MOVW    NEW_DROW, 24(R2)                        ;5635
                                             51 D5 00083             TSTL    DELTA                                   ;5636
                                             5D 15 00085             BLEQ    14$
                          0C    A2           51 A2 00087             SUBW2   DELTA, 12(R2)                           ;5639
                                50      28   A2 D0 0008B             MOVL    40(R2), WSPTR                           ;5640
                                             53 D4 0008F             CLRL    I                                       ;5641
                                             04 11 00091             BRB     8$
                                50      04   A0 D0 00093 7$:         MOVL    4(WSPTR), WSPTR                         ;5642
                    F8          53           51 F3 00097 8$:         AOBLEQ  DELTA, I, 7$
                                             5C 11 0009B             BRB     17$                                     ;5644
                                52      04   AC D0 0009D 9$:         MOVL    DISPID, R2                              ;5658
                                03      08   A2 91 000A1             CMPB    8(R2), #3
                                             0A 12 000A5             BNEQ    11$
                                             0C BB 000A7             PUSHR   #^M<R2,R3>                              ;5660
                          0000V CF           02 FB 000A9             CALLS   #2, DBG$SCR_SCROLL_SOURCE_DOWN
                                7B           50 E8 000AE 10$:        BLBS    R0, 21$
                                51      1E   A2 3C 000B1 11$:        MOVZWL  30(R2), R1                              ;5663
                                50      12   A2 3C 000B5             MOVZWL  18(R2), R0
                                51           50 C2 000B9             SUBL2   R0, R1
                                50      01   A1 9E 000BC             MOVAB   1(R1), R0
                                51      18   A2 3C 000C0             MOVZWL  24(R2), R1                              ;5664
                                51           53 C0 000C4             ADDL2   R3, R1
                                51           50 D1 000C7             CMPL    R0, R1
                                             03 15 000CA             BLEQ    12$
                                50           51 D0 000CC             MOVL    R1, R0
                                             50 D5 000CF 12$:        TSTL    R0                                      ;5662
```

```
                                 03  14 000D1          BGTR    13$                                    ;
                           50    01  D0 000D3          MOVL    #1, R0                                 ;
                           51 18 A2  3C 000D6 13$:      MOVZWL  24(R2), DELTA                          ; 5665
                     51    50    51  C3 000DA          SUBL3   DELTA, NEW_DROW, DELTA                 ;
                        18 A2       50  B0 000DE          MOVW    NEW_DROW, 24(R2)                       ; 5666
                                    51  D5 000E2          TSTL    DELTA                                  ; 5667
                                    46  15 000E4 14$:      BLEQ    21$                                    ;
                        0C A2       51  A0 000E6          ADDW2   DELTA, 12(R2)                          ; 5670
                           50 28 A2  D0 000EA          MOVL    40(R2), WSPTR                          ; 5671
                                    53  D4 000EE          CLRL    I                                      ; 5672
                                    03  11 000F0          BRB     16$                                    ;
                           50       60  D0 000F2 15$:      MOVL    (WSPTR), WSPTR                         ; 5673
                     F9    53       51  F3 000F5 16$:      AOBLEQ  DELTA, I, 15$                          ;
                        28 A2       50  D0 000F9 17$:      MOVL    WSPTR, 40(R2)                          ; 5675
                                    04  00FFD            RET                                            ; 5615
                           50 04 AC  D0 000FE 18$:      MOVL    DISPID, R0                             ; 5686
                           51 1A A0  3C 00102          MOVZWL  26(R0), R1                             ;
                           51       53  C2 00106          SUBL2   R3, R1                                 ;
                                    1D  14 00109          BGTR    20$                                    ;
                           51       01  D0 0010B          MOVL    #1, R1                                 ;
                                    18  11 0010E          BRB     20$                                    ;
                           50 04 AC  D0 00110 19$:      MOVL    DISPID, R0                             ; 5694
                           51 1A A0  3C 00114          MOVZWL  26(R0), R1                             ;
                           51       53  C0 00118          ADDL2   R3, R1                                 ;
                     00000085 8F       51  D1 0011B          CMPL    R1, #133                               ;
                                    04  15 00122          BLEQ    20$                                    ;
                           51 85 8F  9A 00124          MOVZBL  #133, R1                               ;
                        1A A0       51  B0 00128 20$:      MOVW    R1, 26(R0)                             ;
                                    04  0012C 21$:      RET                                            ; 5709

; Routine Size:  301 bytes,    Routine Base:  DBG$CODE + 30D8
```

```
;5621   5710  1 ROUTINE DBG$SCR_SCROLL_SOURCE_DOWN(DISPID, AMOUNT) =
;5622   5711  1 !
;5623   5712  1 ! FUNCTION
;5624   5713  1 !       This routine scrolls the window down over a source display a specified
;5625   5714  1 !       number of lines.  It accepts as input a pointer to the Screen Display
;5626   5715  1 !       Entry of a display to be scrolled down and the amount by which it is to
;5627   5716  1 !       be scrolled.  It then determines if the display can be scrolled entire-
;5628   5717  1 !       ly "in memory", i.e. within the Display Line Entries already attached
;5629   5718  1 !       to the display.  If it can, this routine returns FALSE and lets the
;5630   5719  1 !       caller (namely DBG$SCR_SCROLL_DISPLAY) do the actual scrolling.
;5631   5720  1 !
;5632   5721  1 !       If the display cannot be scrolled "in memory", this routine discards
;5633   5722  1 !       the current display contents, computes a new central line number, and
;5634   5723  1 !       calls DBG$SRC_TYPE_LNUM_SOURCE to read in a range of source lines
;5635   5724  1 !       around that central line number.  It does the necessary bookkeeping
;5636   5725  1 !       to allow a scrolling count to be computed for the display.  After the
;5637   5726  1 !       display has been filled with the new source lines, this routine returns
;5638   5727  1 !       TRUE to indicate that the scrolling operation has been fully completed.
;5639   5728  1 !
;5640   5729  1 ! INPUTS
;5641   5730  1 !       DISPID  - The Display ID (a pointer to the Screen Display Entry) of
;5642   5731  1 !                 the source display to be scrolled.
;5643   5732  1 !
;5644   5733  1 !       AMOUNT  - The number of lines by which the DISPID display is to be
;5645   5734  1 !                 scrolled down.  This number must be zero or positive.
;5646   5735  1 !
;5647   5736  1 ! OUTPUTS
;5648   5737  1 !       If this routine scrolled the DISPID display, then TRUE is returned as
;5649   5738  1 !           the routine value.  If the DISPID display was not scrolled
;5650   5739  1 !           because there is no need to read in more source lines into
;5651   5740  1 !           the display, then FALSE is returned as the routine value.
;5652   5741  1 !
;5653   5742  1
;5654   5743  2     BEGIN
;5655   5744  2
;5656   5745  2     MAP
;5657   5746  2         DISPID: REF DBG$DISP_ENTRY;     ! Pointer to Display Entry to scroll
;5658   5747  2
;5659   5748  2     LOCAL
;5660   5749  2         CENTRAL_LINE,                   ! Central line number for new source
;5661   5750  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Screen Display Line Entry
;5662   5751  2         FLINK,                          ! Temporary pointer to Error Line Entry
;5663   5752  2         INVSCR_FLAG,                    ! Saved invalidate-scroll-count flag
;5664   5753  2         SAVED_NEXT_LNUM,                ! Saved value of DBG$SRC_NEXT_LNUM
;5665   5754  2         SAVED_SCREEN_SOURCE: VOLATILE;  ! Saved value of DBG$GL_SCREEN_SOURCE
;5666   5755  2
;5667   5756  2     ENABLE
;5668   5757  2         HANDLER_SCROLL_SOURCE(          ! Condition handler to restore the
;5669   5758  2                 SAVED_SCREEN_SOURCE);   !       value of DBG$GL_SCREEN_SOURCE
;5670   5759  2                                         !       in case of error condition
;5671   5760  2
;5672   5761  2
;5673   5762  2
;5674   5763  2     ! We release all Error Line Entries attached to the display whenever it is
;5675   5764  2     ! scrolled vertically (up or down).
;5676   5765  2     !
;5677   5766  2     DLEPTR = .DISPID[DBG$L_DISP_ERROR_PTR];
```

```
; 5678          5767  2     DISPID[DBG$L_DISP_ERROR_PTR] = 0;
; 5679          5768  2     WHILE .DLEPTR NEQ 0 DO
; 5680          5769  3         BEGIN
; 5681          5770  3         FLINK = .DLEPTR[DBG$L_DLINE_FLINK];
; 5682          5771  3         DBG$REL_MEMORY(.DLEPTR);
; 5683          5772  3         DLEPTR = .FLINK;
; 5684          5773  2         END;
; 5685          5774  2
; 5686          5775  2
; 5687          5776  2     ! If the source display can be scrolled the desired amount entirely within
; 5688          5777  2     ! the Display Line Entries it already has in memory, return immediately
; 5689          5778  2     ! with the value FALSE.  This tells the caller to do the scrolling.
; 5690          5779  2     !
; 5691          5780  2     IF .DISPID[DBG$V_DISP_ATBOT] THEN RETURN FALSE;
; 5692          5781  2     IF (.DISPID[DBG$W_DISP_DROW] + .AMOUNT) LEQ
; 5693          5782  3                 (.DISPID[DBG$W_DISP_LINECNT] - .DISPID[DBG$W_DISP_RLEN] + 1)
; 5694          5783  2     THEN
; 5695          5784  2         RETURN FALSE;
; 5696          5785  2
; 5697          5786  2     IF .DISPID[DBG$L_DISP_WINDOW_PTR] EQL 0 THEN RETURN FALSE;
; 5698          5787  2
; 5699          5788  2
; 5700          5789  2     ! The display will have to be scrolled by reading in a new range of line
; 5701          5790  2     ! numbers centered around the line to which the user wants to scroll.
; 5702          5791  2     ! Compute the new central line number around which we will request a range
; 5703          5792  2     ! of source lines.
; 5704          5793  2     !
; 5705          5794  2     DLEPTR = .DISPID[DBG$L_DISP_WINDOW_PTR];
; 5706          5795  2     IF .DLEPTR[DBG$L_DLINE_LINUM] LSS 0
; 5707          5796  2     THEN
; 5708          5797  2         DLEPTR = WINDOW_SOURCE_LINE(.DISPID, TRUE);
; 5709          5798  2
; 5710          5799  2     CENTRAL_LINE = MAX(.DISPID[DBG$L_DISP_MINLINE],
; 5711          5800  2                        MIN(.DISPID[DBG$L_DISP_MAXLINE],
; 5712          5801  2                            .DLEPTR[DBG$L_DLINE_LINUM] +
; 5713          5802  2                            (.DISPID[DBG$W_DISP_RLEN] - 1)/2 + .AMOUNT));
; 5714          5803  2
; 5715          5804  2
; 5716          5805  2     ! If CENTRAL_LINE is the very last line of the module's source text, set
; 5717          5806  2     ! the SCROLL_TO_BOTTOM flag for the DBG$SCR_SOURCE_END routine.
; 5718          5807  2     !
; 5719          5808  2     SCROLL_TO_BOTTOM = FALSE;
; 5720          5809  2     IF .CENTRAL_LINE EQL .DISPID[DBG$L_DISP_MAXLINE]
; 5721          5810  2     THEN
; 5722          5811  2         SCROLL_TO_BOTTOM = TRUE;
; 5723          5812  2
; 5724          5813  2
; 5725          5814  2     ! Now read in a new range of source lines centered around the new central
; 5726          5815  2     ! line number.  Routines DBG$SCR_SOURCE_BEGIN, DBG$SCR_SOURCE_LINE, and
; 5727          5816  2     ! DBG$SCR_SOURCE_END will do the actual scrolling of the source display.
; 5728          5817  2     ! Note that we preserve DBG$SRC_NEXT_LNUM (the next line number for some
; 5729          5818  2     ! of the source display commands) over a scrolling operation.
; 5730          5819  2     !
; 5731          5820  2     SCROLL_AMOUNT = .AMOUNT;
; 5732          5821  2     FIXED_SCROLL_AMOUNT = TRUE;
; 5733          5822  2     FIXED_SCROLL_VALID = TRUE;
; 5734          5823  2     MARKLINE_DISABLE_FLAG = TRUE;
```

```
;  5735          5824  2     SAVED_NEXT_LNUM = .DBG$SRC_NEXT_LNUM;
;  5736          5825  2     SAVED_SCREEN_SOURCE = .DBG$GL_SCREEN_SOURCE;
;  5737          5826  2     DBG$GL_SCREEN_SOURCE = .DISPID;
;  5738          5827  2     DBG$SRC_TYPE_LNUM_SOURCE(.DISPID[DBG$L_DISP_MODPTR],
;  5739          5828  2                              .CENTRAL_LINE, 0, .CENTRAL_LINE, 0, FALSE, FALSE);
;  5740          5829  2     DBG$GL_SCREEN_SOURCE = .SAVED_SCREEN_SOURCE;
;  5741          5830  2     DBG$SRC_NEXT_LNUM = .SAVED_NEXT_LNUM;
;  5742          5831  2
;  5743          5832  2
;  5744          5833  2     ! We are all done scrolling the source display.  Now return.
;  5745          5834  2     !
;  5746          5835  2     RETURN TRUE;
;  5747          5836  2
;  5748          5837  1     END;


                                00FC 00000 DBG$SCR_SCROLL_SOURCE_DOWN:
                                           .WORD   Save R2,R3,R4,R5,R6,R7                  ; 5710
                57 00000000G    00    9E 00002   MOVAB   DBG$SRC_NEXT_LNUM, R7            ;
                56 00000000'    EF    9E 00009   MOVAB   DBG$GL_SCREEN_SOURCE, R6         ;
                55 00000000'    EF    9E 00010   MOVAB   SCROLL_TO_BOTTOM, R5            ;
                                7E    D4 00017   CLRL    SAVED_SCREEN_SOURCE              ; 5743
                6D      00D2    CF    DE 00019   MOVAL   10$, (FP)                       ;
                52        04    AC    D0 0001E   MOVL    DISPID, R2                      ; 5766
                53        2C    A2    D0 00022   MOVL    44(R2), DLEPTR                  ;
                          2C    A2    D4 00026   CLRL    44(R2)                          ; 5767
                                53    D5 00029 1$:  TSTL    DLEPTR                       ; 5768
                                11    13 0002B   BEQL    2$                              ;
                54                    63    D0 0002D   MOVL    (DLEPTR), FLINK          ; 5770
                                      53    DD 00030   PUSHL   DLEPTR                   ; 5771
                00000000G       00          01    FB 00032   CALLS   #1, DBG$REL_MEMORY   ;
                53                          54    D0 00039   MOVL    FLINK, DLEPTR       ; 5772
                                            EB    11 0003C   BRB     1$                  ; 5768
          03        0A    A2                03    E1 0003E 2$:  BBC     #3, 10(R2), 4$    ; 5780
                                          00A6    31 00043 3$:  BRW     9$                ;
                51              18    A2    3C 00046 4$:  MOVZWL  24(R2), R1            ; 5781
                51              08    AC    C0 0004A   ADDL2   AMOUNT, R1               ;
                50              1E    A2    3C 0004E   MOVZWL  30(R2), R0               ; 5782
                54              12    A2    3C 00052   MOVZWL  18(R2), R4               ;
                50                    54    C2 00056   SUBL2   R4, R0                   ;
                                      50    D6 00059   INCL    R0                       ;
                50                    51    D1 0005B   CMPL    R1, R0                   ;
                                      E3    15 0005E   BLEQ    3$                       ;
                                28    A2    D5 00060   TSTL    40(R2)                   ; 5786
                                      DE    13 00063   BEQL    3$                       ;
                53              28    A2    D0 00065   MOVL    40(R2), DLEPTR           ; 5794
                                10    A3    D5 00069   TSTL    16(DLEPTR)               ; 5795
                                      0C    18 0006C   BGEQ    5$                       ;
                                      01    DD 0006E   PUSHL   #1                       ; 5797
                                      52    DD 00070   PUSHL   R2                       ;
                      0000V     CF          02    FB 00072   CALLS   #2, WINDOW_SOURCE_LINE ;
                53                          50    D0 00077   MOVL    R0, DLEPTR        ;
                51              12    A2    3C 0007A 5$:  MOVZWL  18(R2), R1            ; 5802
                                      51    D7 0007E   DECL    R1                       ;
```

```
                        51          02  C6 00080          DIVL2   #2, R1                                          ;
                        51      10  A3  C0 00083          ADDL2   16(DLEPTR), R1                                  ;
                        51      08  AC  C0 00087          ADDL2   AMOUNT, R1                                      ;
                        50      44  A2  D0 0008B          MOVL    68(R2), R0                                      ;
                        51          50  D1 0008F          CMPL    R0, R1                                          ;
                                    03  15 00092          BLEQ    6$                                              ;
                        50          51  D0 00094          MOVL    R1, R0                                          ;
                        51      40  A2  D0 00097 6$:      MOVL    64(R2), R1                                      ; 5800
                        50          51  D1 0009B          CMPL    R1, R0                                          ;
                                    03  18 0009E          BGEQ    7$                                              ;
                        51          50  D0 000A0          MOVL    R0, R1                                          ;
                        50          51  D0 000A3 7$:      MOVL    R1, CENTRAL_LINE                                ; 5799
                                    65  D4 000A6          CLRL    SCROLL_TO_BOTTOM                                ; 5808
                     44 A2          50  D1 000A8          CMPL    CENTRAL_LINE, 68(R2)                            ; 5809
                                    03  12 000AC          BNEQ    8$                                              ;
                        65          01  D0 000AE          MOVL    #1, SCROLL_TO_BOTTOM                            ; 5811
                     FC A5      08  AC  D0 000B1 8$:      MOVL    AMOUNT, SCROLL_AMOUNT                           ; 5820
                   E8D0 C5          01  D0 000B6          MOVL    #1, FIXED_SCROLL_AMOUNT                         ; 5821
                   E8D4 C5          01  D0 000BB          MOVL    #1, FIXED_SCROLL_VALID                          ; 5822
                   E8DC C5          01  D0 000C0          MOVL    #1, MARKLINE_DISABLE_FLAG                       ; 5823
                        53          67  D0 000C5          MOVL    DBG$SRC_NEXT_LNUM, SAVED_NEXT_LNUM              ; 5824
                        6E          66  D0 000C8          MOVL    DBG$GL_SCREEN_SOURCE, SAVED_SCREEN_SOURCE       ; 5825
                        66          52  D0 000CB          MOVL    R2, DBG$GL_SCREEN_SOURCE                        ; 5826
                                    7E  7C 000CE          CLRQ    -(SP)                                           ; 5827
                                    7E  D4 000D0          CLRL    -(SP)                                           ;
                                    50  DD 000D2          PUSHL   CENTRAL_LINE                                    ; 5828
                                    7E  D4 000D4          CLRL    -(SP)                                           ; 5827
                                    50  DD 000D6          PUSHL   CENTRAL_LINE                                    ; 5828
                                34  A2  DD 000D8          PUSHL   52(R2)                                          ; 5827
                00000000G       00      07  FB 000DB          CALLS   #7, DBG$SRC_TYPE_LNUM_SOURCE                    ;
                                66      6E  D0 000E2          MOVL    SAVED_SCREEN_SOURCE, DBG$GL_SCREEN_SOURCE       ; 5829
                                67      53  D0 000E5          MOVL    SAVED_NEXT_LNUM, DBG$SRC_NEXT_LNUM              ; 5830
                                50      01  D0 000E8          MOVL    #1, R0                                          ; 5835
                                            04  000EB          RET                                                 ;
                                        50  D4 000EC 9$:      CLRL    R0                                              ; 5837
                                            04  000EE          RET                                                 ;
                                      0000  000EF 10$:      .WORD   Save nothing                                    ; 5743
                                50      08  AC  D0 000F1          MOVL    8(AP), R0                               ;
                                50      04  A0  D0 000F5          MOVL    4(R0), R0                               ;
                                        FC  A0  9F 000F9          PUSHAB  SAVED_SCREEN_SOURCE                     ;
                                            01  DD 000FC          PUSHL   #1                                      ;
                                            5E  DD 000FE          PUSHL   SP                                      ;
                                7E      04  AC  7D 00100          MOVQ    4(AP), -(SP)                            ;
                          0000V CF              03  FB 00104          CALLS   #3, HANDLER_SCROLL_SOURCE               ;
                                                04  00109          RET                                             ;

; Routine Size:  266 bytes,    Routine Base:  DBG$CODE + 3205
```

```
;  5750          5838  1 ROUTINE DBG$SCR_SCROLL_SOURCE_UP(DISPID, AMOUNT) =
;  5751          5839  1 !
;  5752          5840  1 ! FUNCTION
;  5753          5841  1 !       This routine scrolls the window up over a source display a specified
;  5754          5842  1 !       number of lines.  It accepts as input a pointer to the Screen Display
;  5755          5843  1 !       Entry of a display to be scrolled up and the amount by which it is to
;  5756          5844  1 !       be scrolled.  It then determines if the display can be scrolled entire-
;  5757          5845  1 !       ly "in memory", i.e. within the Display Line Entries already attached
;  5758          5846  1 !       to the display.  If it can, this routine returns FALSE and lets the
;  5759          5847  1 !       caller (namely DBG$SCR_SCROLL_DISPLAY) do the actual scrolling.
;  5760          5848  1 !
;  5761          5849  1 !       If the display cannot be scrolled "in memory", this routine discards
;  5762          5850  1 !       the current display contents, computes a new central line number, and
;  5763          5851  1 !       calls DBG$SRC_TYPE_LNUM_SOURCE to read in a range of source lines
;  5764          5852  1 !       around that central line number.  It does the necessary bookkeeping
;  5765          5853  1 !       to allow a scrolling count to be computed for the display.  After the
;  5766          5854  1 !       display has been filled with the new source lines, this routine returns
;  5767          5855  1 !       TRUE to indicate that the scrolling operation has been fully completed.
;  5768          5856  1 !
;  5769          5857  1 ! INPUTS
;  5770          5858  1 !       DISPID  - The Display ID (a pointer to the Screen Display Entry) of
;  5771          5859  1 !                 the source display to be scrolled.
;  5772          5860  1 !
;  5773          5861  1 !       AMOUNT  - The number of lines by which the DISPID display is to be
;  5774          5862  1 !                 scrolled up.  This number must be zero or positive.
;  5775          5863  1 !
;  5776          5864  1 ! OUTPUTS
;  5777          5865  1 !       If this routine scrolled the DISPID display, then TRUE is returned as
;  5778          5866  1 !                 the routine value.  If the DISPID display was not scrolled
;  5779          5867  1 !                 because there is no need to read in more source lines into
;  5780          5868  1 !                 the display, then FALSE is returned as the routine value.
;  5781          5869  1 !
;  5782          5870  1
;  5783          5871  2     BEGIN
;  5784          5872  2
;  5785          5873  2     MAP
;  5786          5874  2         DISPID: REF DBG$DISP_ENTRY;     ! Pointer to Display Entry to scroll
;  5787          5875  2
;  5788          5876  2     LOCAL
;  5789          5877  2         CENTRAL_LINE,                   ! Central line number for new source
;  5790          5878  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Screen Display Line Entry
;  5791          5879  2         FLINK,                          ! Temporary pointer to Error Line Entry
;  5792          5880  2         INVSCR_FLAG,                    ! Saved invalidate-scroll-count flag
;  5793          5881  2         SAVED_NEXT_LNUM,                ! Saved value of DBG$SRC_NEXT_LNUM
;  5794          5882  2         SAVED_SCREEN_SOURCE: VOLATILE;  ! Saved value of DBG$GL_SCREEN_SOURCE
;  5795          5883  2
;  5796          5884  2     ENABLE
;  5797          5885  2         HANDLER_SCROLL_SOURCE(          ! Condition handler to restore the
;  5798          5886  2                 SAVED_SCREEN_SOURCE);   !       value of DBG$GL_SCREEN_SOURCE
;  5799          5887  2                                         !       in case of error condition
;  5800          5888  2
;  5801          5889  2
;  5802          5890  2
;  5803          5891  2     ! We release all Error Line Entries attached to the display whenever it is
;  5804          5892  2     ! scrolled vertically (up or down).
;  5805          5893  2     !
;  5806          5894  2     DLEPTR = .DISPID[DBG$L_DISP_ERROR_PTR];
```

```
; 5807          5895  2     DISPID[DBG$L_DISP_ERROR_PTR] = 0;
; 5808          5896  2     WHILE .DLEPTR NEQ 0 DO
; 5809          5897  3         BEGIN
; 5810          5898  3         FLINK = .DLEPTR[DBG$L_DLINE_FLINK];
; 5811          5899  3         DBG$REL_MEMORY(.DLEPTR);
; 5812          5900  3         DLEPTR = .FLINK;
; 5813          5901  2         END;
; 5814          5902  2
; 5815          5903  2
; 5816          5904  2     ! If the source display can be scrolled the desired amount entirely within
; 5817          5905  2     ! the Display Line Entries it already has in memory, return immediately
; 5818          5906  2     ! with the value FALSE.  This tells the caller to do the scrolling.
; 5819          5907  2     !
; 5820          5908  2     IF .DISPID[DBG$V_DISP_ATTOP] THEN RETURN FALSE;
; 5821          5909  2     IF (.DISPID[DBG$W_DISP_DROW] - .AMOUNT) GTR 0 THEN RETURN FALSE;
; 5822          5910  2     IF .DISPID[DBG$L_DISP_WINDOW_PTR] EQL 0 THEN RETURN FALSE;
; 5823          5911  2
; 5824          5912  2
; 5825          5913  2     ! The display will have to be scrolled by reading in a new range of line
; 5826          5914  2     ! numbers centered around the line to which the user wants to scroll.
; 5827          5915  2     ! Compute the new central line number around which we will request a range
; 5828          5916  2     ! of source lines.
; 5829          5917  2     !
; 5830          5918  2     DLEPTR = .DISPID[DBG$L_DISP_WINDOW_PTR];
; 5831          5919  2     IF .DLEPTR[DBG$L_DLINE_LINUM] LSS 0
; 5832          5920  2     THEN
; 5833          5921  2         DLEPTR = WINDOW_SOURCE_LINE(.DISPID, FALSE);
; 5834          5922  2
; 5835          5923  2     CENTRAL_LINE = MAX(.DISPID[DBG$L_DISP_MINLINE],
; 5836          5924  2                        MIN(.DISPID[DBG$L_DISP_MAXLINE],
; 5837          5925  2                            .DLEPTR[DBG$L_DLINE_LINUM] +
; 5838          5926  2                            (.DISPID[DBG$W_DISP_RLEN] - 1)/2 - .AMOUNT));
; 5839          5927  2
; 5840          5928  2
; 5841          5929  2     ! Now read in a new range of source lines centered around the new central
; 5842          5930  2     ! line number.  Routines DBG$SCR_SOURCE_BEGIN, DBG$SCR_SOURCE_LINE, and
; 5843          5931  2     ! DBG$SCR_SOURCE_END will do the actual scrolling of the source display.
; 5844          5932  2     ! Note that we preserve DBG$SRC_NEXT_LNUM (the next line number for some
; 5845          5933  2     ! of the source display commands) over a scrolling operation.
; 5846          5934  2     !
; 5847          5935  2     SCROLL_AMOUNT = -.AMOUNT;
; 5848          5936  2     FIXED_SCROLL_AMOUNT = TRUE;
; 5849          5937  2     FIXED_SCROLL_VALID = TRUE;
; 5850          5938  2     SCROLL_TO_BOTTOM = FALSE;
; 5851          5939  2     MARKLINE_DISABLE_FLAG = TRUE;
; 5852          5940  2     SAVED_NEXT_LNUM = .DBG$SRC_NEXT_LNUM;
; 5853          5941  2     SAVED_SCREEN_SOURCE = .DBG$GL_SCREEN_SOURCE;
; 5854          5942  2     DBG$GL_SCREEN_SOURCE = .DISPID;
; 5855          5943  2     DBG$SRC_TYPE_LNUM_SOURCE(.DISPID[DBG$L_DISP_MODPTR],
; 5856          5944  2                              .CENTRAL_LINE, 0, .CENTRAL_LINE, 0, FALSE, FALSE);
; 5857          5945  2     DBG$GL_SCREEN_SOURCE = .SAVED_SCREEN_SOURCE;
; 5858          5946  2     DBG$SRC_NEXT_LNUM = .SAVED_NEXT_LNUM;
; 5859          5947  2
; 5860          5948  2
; 5861          5949  2     ! We are all done scrolling the source display.  Now return.
; 5862          5950  2     !
; 5863          5951  2     RETURN TRUE;
```

```
;  5864          5952  2
;  5865          5953  1     END;


                                                00FC 00000 DBG$SCR_SCROLL_SOURCE_UP:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7                  ; 5838
                                    57 00000000G   00   9E 00002          MOVAB   DBG$SRC_NEXT_LNUM, R7
                                    56 00000000'   EF   9E 00009          MOVAB   DBG$GL_SCREEN_SOURCE, R6
                                    55 00000000'   EF   9E 00010          MOVAB   SCROLL_AMOUNT, R5
                                                   7E   D4 00017          CLRL    SAVED_SCREEN_SOURCE     ; 5871
                                    6D      00B8   CF   DE 00019          MOVAL   9$, (FP)
                                    52        04   AC   D0 0001E          MOVL    DISPID, R2              ; 5894
                                    53        2C   A2   D0 00022          MOVL    44(R2), DLEPTR
                                              2C   A2   D4 00026          CLRL    44(R2)                  ; 5895
                                                   53   D5 00029 1$:      TSTL    DLEPTR                  ; 5896
                                                   11   13 0002B          BEQL    2$
                                    54                 63   D0 0002D          MOVL    (DLEPTR), FLINK     ; 5898
                                                   53   DD 00030          PUSHL   DLEPTR                  ; 5899
                                 00000000G         00   FB 00032          CALLS   #1, DBG$REL_MEMORY
                                    53                 54   D0 00039          MOVL    FLINK, DLEPTR       ; 5900
                                                   EB   11 0003C          BRB     1$                      ; 5896
                              03          0A       A2   02   E1 0003E 2$:      BBC     #2, 10(R2), 4$      ; 5908
                                                 008C   31 00043 3$:      BRW     8$
      08    AC          18    A2                 10   00   ED 00046 4$:      CMPZV   #0, #16, 24(R2), AMOUNT ; 5909
                                                   F4   14 0004D          BGTR    3$
                                          28       A2   D5 0004F          TSTL    40(R2)                  ; 5910
                                                   7E   13 00052          BEQL    8$
                                    53            28   A2   D0 00054          MOVL    40(R2), DLEPTR      ; 5918
                                                  10   A3   D5 00058          TSTL    16(DLEPTR)          ; 5919
                                                   0C   18 0005B          BGEQ    5$
                                                   7E   D4 0005D          CLRL    -(SP)                   ; 5921
                                                   52   DD 0005F          PUSHL   R2
                                 0000V             CF   02   FB 00061          CALLS   #2, WINDOW_SOURCE_LINE
                                    53                 50   D0 00066          MOVL    R0, DLEPTR
                                    51            12   A2   3C 00069 5$:      MOVZWL  18(R2), R1          ; 5926
                                                   51   D7 0006D          DECL    R1
                                    51                 02   C6 0006F          DIVL2   #2, R1
                                    51            10   A3   C0 00072          ADDL2   16(DLEPTR), R1
                                    51            08   AC   C2 00076          SUBL2   AMOUNT, R1
                                    50            44   A2   D0 0007A          MOVL    68(R2), R0
                                    51                 50   D1 0007E          CMPL    R0, R1
                                                   03   15 00081          BLEQ    6$
                                    50                 51   D0 00083          MOVL    R1, R0
                                    51            40   A2   D0 00086 6$:      MOVL    64(R2), R1          ; 5924
                                    50                 51   D1 0008A          CMPL    R1, R0
                                                   03   18 0008D          BGEQ    7$
                                    51                 50   D0 0008F          MOVL    R0, R1
                                    50                 51   D0 00092 7$:      MOVL    R1, CENTRAL_LINE    ; 5923
                                    65            08   AC   CE 00095          MNEGL   AMOUNT, SCROLL_AMOUNT ; 5935
                              E8D4  C5                 01   D0 00099          MOVL    #1, FIXED_SCROLL_AMOUNT ; 5936
                              E8D8  C5                 01   D0 0009E          MOVL    #1, FIXED_SCROLL_VALID ; 5937
                                              04       A5   D4 000A3          CLRL    SCROLL_TO_BOTTOM    ; 5938
                              E8E0  C5                 01   D0 000A6          MOVL    #1, MARKLINE_DISABLE_FLAG ; 5939
                                    53                 67   D0 000AB          MOVL    DBG$SRC_NEXT_LNUM, SAVED_NEXT_LNUM ; 5940
```

```
                        6E              66  D0 000AE          MOVL    DBG$GL_SCREEN_SOURCE, SAVED_SCREEN_SOURCE    ; 5941
                        66              52  D0 000B1          MOVL    R2, DBG$GL_SCREEN_SOURCE                     ; 5942
                                        7E  7C 000B4          CLRQ    -(SP)                                        ; 5943
                                        7E  D4 000B6          CLRL    -(SP)                                        ;
                                        50  DD 000B8          PUSHL   CENTRAL_LINE                                 ; 5944
                                        7E  D4 000BA          CLRL    -(SP)                                        ; 5943
                                        50  DD 000BC          PUSHL   CENTRAL_LINE                                 ; 5944
                                    34  A2  DD 000BE          PUSHL   52(R2)                                       ; 5943
                00000000G       00      07  FB 000C1          CALLS   #7, DBG$SRC_TYPE_LNUM_SOURCE                 ;
                                66      6E  D0 000C8          MOVL    SAVED_SCREEN_SOURCE, DBG$GL_SCREEN_SOURCE    ; 5945
                                67      53  D0 000CB          MOVL    SAVED_NEXT_LNUM, DBG$SRC_NEXT_LNUM           ; 5946
                                50      01  D0 000CE          MOVL    #1, R0                                       ; 5951
                                            04 000D1          RET                                                  ;
                                50      D4  D0 000D2 8$:      CLRL    R0                                           ; 5953
                                            04 000D4          RET                                                  ;
                                      0000  000D5 9$:         .WORD   Save nothing                                 ; 5871
                                50  08  AC  D0 000D7          MOVL    8(AP), R0                                    ;
                                50  04  A0  D0 000DB          MOVL    4(R0), R0                                    ;
                                    FC  A0  9F 000DF          PUSHAB  SAVED_SCREEN_SOURCE                          ;
                                        01  DD 000E2          PUSHL   #1                                           ;
                                        5E  DD 000E4          PUSHL   SP                                           ;
                                7E  04  AC  7D 000E6          MOVQ    4(AP), -(SP)                                 ;
                          0000V CF          03  FB 000EA      CALLS   #3, HANDLER_SCROLL_SOURCE                    ;
                                            04 000EF          RET                                                  ;

; Routine Size:  240 bytes,    Routine Base:  DBG$CODE + 330F
```

```
;5867    5954  1 GLOBAL ROUTINE DBG$SCR_SOURCE_BEGIN(OLD_LOW, OLD_HIGH,
;5868    5955  1                                     NEW_LOW, NEW_HIGH, MODRSTPTR, ALL_DONE_FLAG): NOVALUE =
;5869    5956  1 !
;5870    5957  1 ! FUNCTION
;5871    5958  1 !       This routine sets up the output of source lines to a SOURCE screen
;5872    5959  1 !       display.  It also computes the range to source lines to get for this
;5873    5960  1 !       display based on the display size.  After this routine is called, calls
;5874    5961  1 !       on DBG$SCR_SOURCE_LINE are used to actually output the individual lines
;5875    5962  1 !       to the current SOURCE display.
;5876    5963  1 !
;5877    5964  1 !       This routine accepts a line number range and a Module RST Entry pointer
;5878    5965  1 !       as input.  The line number range is derived from the source display
;5879    5966  1 !       command which is being intercepted--it usually consists of only a
;5880    5967  1 !       single line.  The output of this routine is a modified line number
;5881    5968  1 !       range which is calculated to produce a suitable number of lines around
;5882    5969  1 !       the central line of the original range, given the size of the screen
;5883    5970  1 !       display.  The Module RST Entry pointer is saved in the selected SOURCE
;5884    5971  1 !       Screen Display Entry and is later used to retrieve the module name.
;5885    5972  1 !
;5886    5973  1 !       In order to prevent unnecessary I/O, this routine checks whether the
;5887    5974  1 !       screen display's Line Entry List already contains the entire range of
;5888    5975  1 !       lines needed to fill the display's screen window.  If it does, the
;5889    5976  1 !       display is simply scrolled (the window pointer, DROW, and scrolling
;5890    5977  1 !       count are computed and set), and TRUE is returned to the ALL_DONE_FLAG
;5891    5978  1 !       parameter.  This tells the caller to omit the reading of source lines.
;5892    5979  1 !
;5893    5980  1 ! INPUTS
;5894    5981  1 !       OLD_LOW - The lower line number of the original line number range
;5895    5982  1 !                 requested by the intercepted source line display command.
;5896    5983  1 !
;5897    5984  1 !       OLD_HIGH - The upper line number of the original line number range
;5898    5985  1 !                  requested by the intercepted source line display command.
;5899    5986  1 !
;5900    5987  1 !       NEW_LOW - The address of a longword location to receive the lower line
;5901    5988  1 !                 number of the recomputed line number range.
;5902    5989  1 !
;5903    5990  1 !       NEW_HIGH - The address of a longword location to receive the upper line
;5904    5991  1 !                  number of the recomputed line number range.
;5905    5992  1 !
;5906    5993  1 !       MODRSTPTR - A pointer to the Module RST Entry of the module from which
;5907    5994  1 !                   source lines are being displayed.
;5908    5995  1 !
;5909    5996  1 !       ALL_DONE_FLAG - The address of a longword location to receive a flag
;5910    5997  1 !                       value saying that the source operation is all done when
;5911    5998  1 !                       this routine returns.
;5912    5999  1 !
;5913    6000  1 ! OUTPUTS
;5914    6001  1 !       NEW_LOW - A recomputed lower line number is returned to the NEW_LOW
;5915    6002  1 !                 location.  This recomputed lower bound will actually be
;5916    6003  1 !                 used when source lines are retrieved from the source files.
;5917    6004  1 !
;5918    6005  1 !       NEW_HIGH - A recomputed upper line number is returned to the NEW_HIGH
;5919    6006  1 !                  location.  This recomputed upper bound will actually be
;5920    6007  1 !                  used when source lines are retrieved from the source files.
;5921    6008  1 !
;5922    6009  1 !       ALL_DONE_FLAG - The value TRUE is returned to ALL_DONE_FLAG if the
;5923    6010  1 !                       desired range of source lines is already in the current
```

```
;  5924          6011  1 !                 source display data structure.  The significance of this
;  5925          6012  1 !                 is that there is no need to read in any more source lines
;  5926          6013  1 !                 since we already have the desired source lines.  FALSE is
;  5927          6014  1 !                 returned if we do not have all needed source lines, in
;  5928          6015  1 !                 which case more must be read in from the source files.
;  5929          6016  1 !
;  5930          6017  1
;  5931          6018  2     BEGIN
;  5932          6019  2
;  5933          6020  2     MAP
;  5934          6021  2         NEW_LOW: REF VECTOR[1,LONG],    ! Pointer to new lower line number
;  5935          6022  2         NEW_HIGH: REF VECTOR[1,LONG],   ! Pointer to new upper line number
;  5936          6023  2         ALL_DONE_FLAG:                  ! Pointer to the all-done-flag which
;  5937          6024  2                 REF VECTOR[1,LONG];     !      says we already have the source
;  5938          6025  2                                         !      lines being requested
;  5939          6026  2
;  5940          6027  2     LOCAL
;  5941          6028  2         CENTER_LINE,                    ! Central line number of display text
;  5942          6029  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to Source Screen Display Entry
;  5943          6030  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Source Display Line Entry
;  5944          6031  2         MAXLINE,                        ! Maximum line number in desired range
;  5945          6032  2         MINLINE,                        ! Minimum line number in desired range
;  5946          6033  2         SCROLL_AMOUNT,                  ! Amount to scroll display if display
;  5947          6034  2                                         !      already has the desired lines
;  5948          6035  2         SIZE;                           ! Number of source lines to get for
;  5949          6036  2                                         !      this display
;  5950          6037  2
;  5951          6038  2     LABEL
;  5952          6039  2         CHECK_HAVE_LINES;               ! Block that checks whether we already
;  5953          6040  2                                         !      have the desired range of lines
;  5954          6041  2
;  5955          6042  2
;  5956          6043  2
;  5957          6044  2     ! Pick up the pointer to the SOURCE Screen Display Entry.  Check it and
;  5958          6045  2     ! the input parameters for validity.
;  5959          6046  2     !
;  5960          6047  2     DISPTR = .DBG$GL_SCREEN_SOURCE;
;  5961          6048  2     IF .DISPTR[DBG$B_DISP_KIND] NEQ DBG$K_DISP_SOURCE
;  5962          6049  2     THEN
;  5963          6050  2         $DBG_ERROR('DBGSCREEN\SOURCE_BEGIN 10');
;  5964          6051  2
;  5965          6052  2     IF .OLD_LOW GTR .OLD_HIGH THEN $DBG_ERROR('DBGSCREEN\SOURCE_BEGIN 20');
;  5966          6053  2
;  5967          6054  2
;  5968          6055  2     ! If we came in because of a TYPE or EXAMINE/SOURCE command, mark the
;  5969          6056  2     ! fixed-scroll-amount flag as false.  This flag should only be true if we
;  5970          6057  2     ! came here because of SCROLL command which requires more source lines
;  5971          6058  2     ! to be read into the scrolled display.  The flag is later used in the
;  5972          6059  2     ! DBG$SCR_SOURCE_END routine to determine the scrolling count.
;  5973          6060  2     !
;  5974          6061  2     IF NOT .FIXED_SCROLL_VALID THEN FIXED_SCROLL_AMOUNT = FALSE;
;  5975          6062  2     FIXED_SCROLL_VALID = FALSE;
;  5976          6063  2
;  5977          6064  2
;  5978          6065  2     ! Check whether we already have the source lines to be display in the
;  5979          6066  2     ! Screen Display Line Entry List for this source diplay.  If so, there
;  5980          6067  2     ! is no need to read them in from the source file.
```

```
; 5981          6068  2     !
; 5982          6069  2     CHECK_HAVE_LINES:
; 5983          6070  2         BEGIN
; 5984          6071  2
; 5985          6072  2
; 5986          6073  2         ! If the source module to be displayed is different from the module we
; 5987          6074  2         ! have in this screen display, we must read in the source lines from
; 5988          6075  2         ! the source file, so we leave CHECK_HAVE_LINES.  Similarly, if the
; 5989          6076  2         ! display does not at present have any lines, we leave.
; 5990          6077  2         !
; 5991          6078  3         IF (.MODRSTPTR NEQ .DISPTR[DBG$L_DISP_MODPTR]) OR
; 5992          6079  4            (.DISPTR[DBG$W_DISP_LINECNT] EQL 0)                                    ;
; 5993          6080  3         THEN
; 5994          6081  3             LEAVE CHECK_HAVE_LINES;
; 5995          6082  3
; 5996          6083  3
; 5997          6084  3         ! Compute the new range of line numbers we want to display in the
; 5998          6085  3         ! screen display's window.
; 5999          6086  3         !
; 6000          6087  3         CENTER_LINE = .OLD_LOW + (.OLD_HIGH - .OLD_LOW)/2;
; 6001          6088  3         MINLINE = MAX(.DISPTR[DBG$L_DISP_MINLINE],
; 6002          6089  3                       .CENTER_LINE - (.DISPTR[DBG$W_DISP_RLEN] - 1)/2);
; 6003          6090  3         MAXLINE = MIN(.DISPTR[DBG$L_DISP_MAXLINE],
; 6004          6091  3                       .MINLINE + .DISPTR[DBG$W_DISP_RLEN] - 1);
; 6005          6092  3         MINLINE = MAX(.DISPTR[DBG$L_DISP_MINLINE],
; 6006          6093  3                       .MAXLINE - .DISPTR[DBG$W_DISP_RLEN] + 1);
; 6007          6094  3
; 6008          6095  3
; 6009          6096  3         ! See if this range of line numbers is already present in the Screen
; 6010          6097  3         ! Display Line Entry List for this screen display.  If not, we leave
; 6011          6098  3         ! CHECK_HAVE_LINES so that the lines are read in from the source file.
; 6012          6099  3         !
; 6013          6100  3         DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
; 6014          6101  3         IF .DLEPTR[DBG$L_DLINE_LINUM] GTR .MINLINE THEN LEAVE CHECK_HAVE_LINES;
; 6015          6102  3         DLEPTR = .DISPTR[DBG$L_DISP_END_LINE_PTR];
; 6016          6103  3         IF .DLEPTR[DBG$L_DLINE_LINUM] LSS .MAXLINE THEN LEAVE CHECK_HAVE_LINES;
; 6017          6104  3
; 6018          6105  3
; 6019          6106  3         ! The desired range of source lines is already in the Screen Display
; 6020          6107  3         ! Line Entry List.  Now determine whether to scroll up or down the
; 6021          6108  3         ! display to get the desired lines in the screen window.  If we must
; 6022          6109  3         ! scroll up we determine the scrolling amount in the up direction by
; 6023          6110  3         ! counting how many lines the window pointer must be moved up.
; 6024          6111  3         !
; 6025          6112  3         SCROLL_AMOUNT = 0;
; 6026          6113  3         DLEPTR = .DISPTR[DBG$L_DISP_WINDOW_PTR];
; 6027          6114  3         IF .DLEPTR[DBG$L_DLINE_LINUM] GEQ .MINLINE
; 6028          6115  3         THEN
; 6029          6116  4             BEGIN
; 6030          6117  4             WHILE .DLEPTR[DBG$L_DLINE_LINUM] GTR .MINLINE DO
; 6031          6118  5                 BEGIN
; 6032          6119  5                 SCROLL_AMOUNT = .SCROLL_AMOUNT - 1;
; 6033          6120  5                 DLEPTR = .DLEPTR[DBG$L_DLINE_BLINK];
; 6034          6121  5                 IF .DLEPTR EQL DISPTR[DBG$L_DISP_START_LINE_PTR]
; 6035          6122  5                 THEN
; 6036          6123  5                     $DBG_ERROR('DBGSCREEN\SOURCE_BEGIN 30');
; 6037          6124  5
```

```
; 6038          6125  4                 END;
; 6039          6126  4
; 6040          6127  4             END
; 6041          6128  4
; 6042          6129  4
; 6043          6130  4         ! Else we must scroll down.  We determine the scrolling amount in the
; 6044          6131  4         ! down direction by counting how many lines the window pointer must be
; 6045          6132  4         ! moved down.
; 6046          6133  4         !
; 6047          6134  3         ELSE
; 6048          6135  4             BEGIN
; 6049          6136  4             WHILE .DLEPTR[DBG$L_DLINE_LINUM] LSS .MINLINE DO
; 6050          6137  5                 BEGIN
; 6051          6138  5                 SCROLL_AMOUNT = .SCROLL_AMOUNT + 1;
; 6052          6139  5                 DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
; 6053          6140  5                 IF .DLEPTR EQL DISPTR[DBG$L_DISP_START_LINE_PTR]
; 6054          6141  5                 THEN
; 6055          6142  5                     $DBG_ERROR('DBGSCREEN\SOURCE_BEGIN 40');
; 6056          6143  5
; 6057          6144  4                 END;
; 6058          6145  4
; 6059          6146  3             END;
; 6060          6147  3
; 6061          6148  3
; 6062          6149  3         ! Now set the updated window pointer, adjust the DROW value (display
; 6063          6150  3         ! row location of start of window), and fill in the scrolling count.
; 6064          6151  3         ! Also set the center line number in the Display Entry.
; 6065          6152  3         !
; 6066          6153  3         DISPTR[DBG$L_DISP_WINDOW_PTR] = .DLEPTR;
; 6067          6154  3         DISPTR[DBG$W_DISP_DROW] = .DISPTR[DBG$W_DISP_DROW] + .SCROLL_AMOUNT;
; 6068          6155  3         DISPTR[DBG$W_DISP_SCROLL] = .DISPTR[DBG$W_DISP_SCROLL] + .SCROLL_AMOUNT;
; 6069          6156  3         DISPTR[DBG$L_DISP_CENTER] = MAX(.DISPTR[DBG$L_DISP_MINLINE],
; 6070          6157  3                         MIN(.DISPTR[DBG$L_DISP_MAXLINE], .CENTER_LINE));
; 6071          6158  3
; 6072          6159  3
; 6073          6160  3         ! Set the next line number (as used by the parameter-less TYPE and
; 6074          6161  3         ! SEARCH commands) to be the specified center line plus one.
; 6075          6162  3         !
; 6076          6163  3         DBG$SRC_NEXT_LNUM = .DISPTR[DBG$L_DISP_CENTER] + 1;
; 6077          6164  3
; 6078          6165  3
; 6079          6166  3         ! Unless the user is typing a range of line numbers (as opposed to a
; 6080          6167  3         ! single line) and unless marking is disabled because we are simply
; 6081          6168  3         ! scrolling, mark the central line as the line to be marked with a
; 6082          6169  3         ! "->" in the display.
; 6083          6170  3         !
; 6084          6171  3         DISPTR[DBG$L_DISP_MARKLINE] = -1;
; 6085          6172  4         IF (.OLD_LOW EQL .OLD_HIGH) AND (NOT .MARKLINE_DISABLE_FLAG)
; 6086          6173  3         THEN
; 6087          6174  3             DISPTR[DBG$L_DISP_MARKLINE] = .CENTER_LINE;
; 6088          6175  3
; 6089          6176  3         MARKLINE_DISABLE_FLAG = FALSE;
; 6090          6177  3
; 6091          6178  3
; 6092          6179  3         ! The screen display is now positioned correctly at the desired range
; 6093          6180  3         ! of source lines.  Since there is no further need to read in any
; 6094          6181  3         ! source lines, we set the ALL_DONE_FLAG and return to the caller.
```

```
;  6095     6182  3          !
;  6096     6183  3          ALL_DONE_FLAG[0] = TRUE;
;  6097     6184  3          RETURN;
;  6098     6185  3
;  6099     6186  2          END;                                ! End of CHECK_HAVE_LINES block
;  6100     6187  2
;  6101     6188  2
;  6102     6189  2      ! We do not yet have the desired source lines in the Display Line Entry
;  6103     6190  2      ! List.  Hence we set the ALL_DONE_FLAG to FALSE.
;  6104     6191  2      !
;  6105     6192  2      ALL_DONE_FLAG[0] = FALSE;
;  6106     6193  2
;  6107     6194  2
;  6108     6195  2      ! Save all information from the current contents of the Screen Display
;  6109     6196  2      ! Entry that we will need to compute a scrolling count at the end of
;  6110     6197  2      ! the source retrieval operation.
;  6111     6198  2      !
;  6112     6199  2      INVSCR_FLAG = .DISPTR[DBG$V_DISP_INVSCR];
;  6113     6200  3      IF (NOT .INVSCR_FLAG) AND (.DISPTR[DBG$W_DISP_LINECNT] GTR 0)
;  6114     6201  2      THEN
;  6115     6202  3          BEGIN
;  6116     6203  3          SAVED_SCROLL = .DISPTR[DBG$W_DISP_SCROLL];
;  6117     6204  3          TOP_OFFSET = .DISPTR[DBG$W_DISP_DROW] - 1;
;  6118     6205  3          BOT_OFFSET = .DISPTR[DBG$W_DISP_LINECNT] - .DISPTR[DBG$W_DISP_DROW];
;  6119     6206  3          DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  6120     6207  3          TOP_REF_FILEID = .DLEPTR[DBG$W_DLINE_FILEID];
;  6121     6208  3          TOP_REF_RECNUM = .DLEPTR[DBG$L_DLINE_RECNUM];
;  6122     6209  3          DLEPTR = .DISPTR[DBG$L_DISP_END_LINE_PTR];
;  6123     6210  3          BOT_REF_FILEID = .DLEPTR[DBG$W_DLINE_FILEID];
;  6124     6211  3          BOT_REF_RECNUM = .DLEPTR[DBG$L_DLINE_RECNUM];
;  6125     6212  2          END;
;  6126     6213  2
;  6127     6214  2
;  6128     6215  2      ! Empty the specified display of all its current contents.
;  6129     6216  2      !
;  6130     6217  2      DBG$SCR_EMPTY_DISPLAY(.DISPTR);
;  6131     6218  2
;  6132     6219  2
;  6133     6220  2      ! If the source text is coming from a new module than before, set the new
;  6134     6221  2      ! Module RST Entry pointer in the Display Entry and invalidate the scroll-
;  6135     6222  2      ! ing count.  We also initialize all source line number information for the
;  6136     6223  2      ! display, including the line number range of the module.
;  6137     6224  2      !
;  6138     6225  2      IF .MODRSTPTR NEQ .DISPTR[DBG$L_DISP_MODPTR]
;  6139     6226  2      THEN
;  6140     6227  3          BEGIN
;  6141     6228  3          DISPTR[DBG$L_DISP_MODPTR] = .MODRSTPTR;
;  6142     6229  3          DISPTR[DBG$V_DISP_INVSCR] = TRUE;
;  6143     6230  3          DISPTR[DBG$L_DISP_MARKLINE] = -1;
;  6144     6231  3          DBG$SRC_LNUM_RANGE(.MODRSTPTR,
;  6145     6232  3                  DISPTR[DBG$L_DISP_MINLINE], DISPTR[DBG$L_DISP_MAXLINE]);
;  6146     6233  3          INVSCR_FLAG = TRUE;
;  6147     6234  2          END;
;  6148     6235  2
;  6149     6236  2
;  6150     6237  2      ! Compute the central line number of the original line number range and
;  6151     6238  2      ! then compute the new line number range to retrieve, given the size of
```

```
;  6152          6239  2     ! the current display.  Save the central line number in the Display Entry.
;  6153          6240  2     !
;  6154          6241  2     DISPTR[DBG$V_DISP_ATTOP] = FALSE;
;  6155          6242  2     DISPTR[DBG$V_DISP_ATBOT] = FALSE;
;  6156          6243  2     CENTER_LINE = .OLD_LOW + (.OLD_HIGH - .OLD_LOW)/2;
;  6157          6244  2     SIZE = MAX(.DISPTR[DBG$W_DISP_RLEN], .DISPTR[DBG$W_DISP_MAX_LINECNT]);
;  6158          6245  2     NEW_LOW[0] = .CENTER_LINE - (.SIZE - 1)/2;
;  6159          6246  2     NEW_HIGH[0] = .CENTER_LINE + .SIZE/2;
;  6160          6247  2     IF .NEW_LOW[0] LSS .DISPTR[DBG$L_DISP_MINLINE]
;  6161          6248  2     THEN
;  6162          6249  3         BEGIN
;  6163          6250  3         NEW_LOW[0] = .DISPTR[DBG$L_DISP_MINLINE];
;  6164          6251  3         NEW_HIGH[0] = .NEW_LOW[0] + (2*.SIZE - 1)/2;
;  6165          6252  3         DISPTR[DBG$V_DISP_ATTOP] = TRUE;
;  6166          6253  2         END;
;  6167          6254  2
;  6168          6255  2     IF .NEW_HIGH[0] GTR .DISPTR[DBG$L_DISP_MAXLINE]
;  6169          6256  2     THEN
;  6170          6257  3         BEGIN
;  6171          6258  3         NEW_HIGH[0] = .DISPTR[DBG$L_DISP_MAXLINE];
;  6172          6259  3         NEW_LOW[0] = MAX(.DISPTR[DBG$L_DISP_MINLINE],
;  6173          6260  3                          .NEW_HIGH[0] - (2*.SIZE - 1)/2);
;  6174          6261  3         DISPTR[DBG$V_DISP_ATBOT] = TRUE;
;  6175          6262  2         END;
;  6176          6263  2
;  6177          6264  2     DISPTR[DBG$L_DISP_CENTER] = MAX(.DISPTR[DBG$L_DISP_MINLINE],
;  6178          6265  2                                     MIN(.DISPTR[DBG$L_DISP_MAXLINE], .CENTER_LINE));
;  6179          6266  2
;  6180          6267  2
;  6181          6268  2     ! Unless the user is typing a range of line numbers (as opposed to a single
;  6182          6269  2     ! line) and unless marking is disabled because we are simply scrolling,
;  6183          6270  2     ! mark the central line as the line to be marked with a "->" in the display.
;  6184          6271  2     !
;  6185          6272  3     IF (.OLD_LOW EQL .OLD_HIGH) AND (NOT .MARKLINE_DISABLE_FLAG)
;  6186          6273  2     THEN
;  6187          6274  2         DISPTR[DBG$L_DISP_MARKLINE] = .CENTER_LINE;
;  6188          6275  2
;  6189          6276  2     MARKLINE_DISABLE_FLAG = FALSE;
;  6190          6277  2     RETURN;
;  6191          6278  2
;  6192          6279  1     END;


                                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

52 55 4F 53 5C 4E 45 45 52 43 53 47 42 44 19 0078C P.AHW:  .ASCII  <25>\DBGSCREEN\<92>\SOURCE_BEGIN 10\      ;
                30 31 20 4E 49 47 45 42 5F 45 43 0079B                                                      ;
52 55 4F 53 5C 4E 45 45 52 43 53 47 42 44 19 007A6 P.AHX:  .ASCII  <25>\DBGSCREEN\<92>\SOURCE_BEGIN 20\      ;
                30 32 20 4E 49 47 45 42 5F 45 43 007B5                                                      ;
52 55 4F 53 5C 4E 45 45 52 43 53 47 42 44 19 007C0 P.AHY:  .ASCII  <25>\DBGSCREEN\<92>\SOURCE_BEGIN 30\      ;
                30 33 20 4E 49 47 45 42 5F 45 43 007CF                                                      ;
52 55 4F 53 5C 4E 45 45 52 43 53 47 42 44 19 007DA P.AHZ:  .ASCII  <25>\DBGSCREEN\<92>\SOURCE_BEGIN 40\      ;
                30 34 20 4E 49 47 45 42 5F 45 43 007E9                                                      ;
```

```
                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                 0FFC 00000     .ENTRY  DBG$SCR_SOURCE_BEGIN, Save R2,R3,R4,R5,R6,-  ; 5954
                                                        R7,R8,R9,R10,R11                              ;
                5B 00000000'   EF   9E 00002    MOVAB   P.AHW, R11                                    ;
                5A 00000000'   EF   9E 00009    MOVAB   MARKLINE_DISABLE_FLAG, R10                    ;
                52 00000000'   EF   D0 00010    MOVL    DBG$GL_SCREEN_SOURCE, DISPTR                  ; 6047
                58        08   A2   9E 00017    MOVAB   8(DISPTR), R8                                 ; 6048
                03             68   91 0001B    CMPB    (R8), #3                                      ;
                               11   13 0001E    BEQL    1$                                            ;
                               5B   DD 00020    PUSHL   R11                                           ; 6050
                               01   DD 00022    PUSHL   #1                                            ;
                   00028362    8F   DD 00024    PUSHL   #164706                                       ;
00000000G       00             03   FB 0002A    CALLS   #3, LIB$SIGNAL                                ;
                57        04   AC   D0 00031 1$: MOVL    OLD_LOW, R7                                   ; 6052
                59        08   AC   D0 00035    MOVL    OLD_HIGH, R9                                  ;
                59             57   D1 00039    CMPL    R7, R9                                        ;
                               12   15 0003C    BLEQ    2$                                            ;
                            1A AB   9F 0003E    PUSHAB  P.AHX                                         ;
                               01   DD 00041    PUSHL   #1                                            ;
                   00028362    8F   DD 00043    PUSHL   #164706                                       ;
00000000G       00             03   FB 00049    CALLS   #3, LIB$SIGNAL                                ;
                03        F8   AA   E8 00050 2$: BLBS    FIXED_SCROLL_VALID, 3$                        ; 6061
                          F4   AA   D4 00054    CLRL    FIXED_SCROLL_AMOUNT                           ;
                          F8   AA   D4 00057 3$: CLRL    FIXED_SCROLL_VALID                            ; 6062
             34   A2        14   AC   D1 0005A    CMPL    MODRSTPTR, 52(DISPTR)                         ; 6078
                               03   13 0005F    BEQL    5$                                            ;
                            0110    31 00061 4$: BRW     15$                                           ;
                         1E   A2   B5 00064 5$: TSTW    30(DISPTR)                                    ; 6079
                               F8   13 00067    BEQL    4$                                            ;
          50             59        57   C3 00069    SUBL3   R7, R9, R0                                ; 6087
                         50        02   C6 0006D    DIVL2   #2, R0                                    ;
          53             50        57   C1 00070    ADDL3   R7, R0, CENTER_LINE                       ;
                         54   12   A2   3C 00074    MOVZWL  18(DISPTR), R4                            ; 6089
                         50   FF   A4   9E 00078    MOVAB   -1(R4), R0                                ;
                         50        02   C6 0007C    DIVL2   #2, R0                                    ;
          50             53        50   C3 0007F    SUBL3   R0, CENTER_LINE, R0                       ;
                         51   40   A2   D0 00083    MOVL    64(DISPTR), R1                            ;
                         50        51   D1 00087    CMPL    R1, R0                                    ;
                                   03   18 0008A    BGEQ    6$                                        ;
                         51        50   D0 0008C    MOVL    R0, R1                                    ;
                         55        51   D0 0008F 6$: MOVL    R1, MINLINE                               ; 6088
                         50   FF A445   9E 00092    MOVAB   -1(R4)[MINLINE], R0                       ; 6091
                         51   44   A2   D0 00097    MOVL    68(DISPTR), R1                            ;
                         50        51   D1 0009B    CMPL    R1, R0                                    ;
                                   03   15 0009E    BLEQ    7$                                        ;
                         51        50   D0 000A0    MOVL    R0, R1                                    ;
                         50        51   D0 000A3 7$: MOVL    R1, MAXLINE                               ; 6090
          54             50        54   C3 000A6    SUBL3   R4, MAXLINE, R4                           ; 6093
                                   54   D6 000AA    INCL    R4                                        ;
                         51   40   A2   D0 000AC    MOVL    64(DISPTR), R1                            ;
                         54        51   D1 000B0    CMPL    R1, R4                                    ;
                                   03   18 000B3    BGEQ    8$                                        ;
                         51        54   D0 000B5    MOVL    R4, R1                                    ;
                         55        51   D0 000B8 8$: MOVL    R1, MINLINE                               ; 6092
                         54   20   A2   D0 000BB    MOVL    32(DISPTR), DLEPTR                        ; 6100
                         55   10   A4   D1 000BF    CMPL    16(DLEPTR), MINLINE                       ; 6101
```

```
                                 9C  14 000C3          BGTR    4$                                          ;
                          54      24 A2  D0 000C5          MOVL    36(DISPTR), DLEPTR                      ; 6102
                          50      10 A4  D1 000C9          CMPL    16(DLEPTR), MAXLINE                     ; 6103
                                     92  19 000CD          BLSS    4$                                      ;
                                     56  D4 000CF          CLRL    SCROLL_AMOUNT                           ; 6112
                          54      28 A2  D0 000D1          MOVL    40(DISPTR), DLEPTR                      ; 6113
                          55      10 A4  D1 000D5          CMPL    16(DLEPTR), MINLINE                     ; 6114
                                     29  19 000D9          BLSS    10$                                     ;
                          55      10 A4  D1 000DB 9$:      CMPL    16(DLEPTR), MINLINE                     ; 6117
                                     4B  15 000DF          BLEQ    11$                                     ;
                                     56  D7 000E1          DECL    SCROLL_AMOUNT                           ; 6119
                          54      04 A4  D0 000E3          MOVL    4(DLEPTR), DLEPTR                       ; 6120
                          50      20 A2  9E 000E7          MOVAB   32(DISPTR), R0                          ; 6121
                          50         54  D1 000EB          CMPL    DLEPTR, R0                              ;
                                     EB  12 000EE          BNEQ    9$                                      ;
                                 34  AB  9F 000F0          PUSHAB  P.AHY                                   ; 6123
                                     01  DD 000F3          PUSHL   #1                                      ;
                              00028362   8F  DD 000F5          PUSHL   #164706                             ;
                00000000G     00         03  FB 000FB          CALLS   #3, LIB$SIGNAL                      ;
                                     D7  11 00102          BRB     9$                                      ; 6117
                          55      10 A4  D1 00104 10$:     CMPL    16(DLEPTR), MINLINE                     ; 6136
                                     22  18 00108          BGEQ    11$                                     ;
                                     56  D6 0010A          INCL    SCROLL_AMOUNT                           ; 6138
                          54         64  D0 0010C          MOVL    (DLEPTR), DLEPTR                        ; 6139
                          50      20 A2  9E 0010F          MOVAB   32(DISPTR), R0                          ; 6140
                          50         54  D1 00113          CMPL    DLEPTR, R0                              ;
                                     EC  12 00116          BNEQ    10$                                     ;
                                 4E  AB  9F 00118          PUSHAB  P.AHZ                                   ; 6142
                                     01  DD 0011B          PUSHL   #1                                      ;
                              00028362   8F  DD 0011D          PUSHL   #164706                             ;
                00000000G     00         03  FB 00123          CALLS   #3, LIB$SIGNAL                      ;
                                     D8  11 0012A          BRB     10$                                     ; 6136
                       28    A2         54  D0 0012C 11$:     MOVL    DLEPTR, 40(DISPTR)                      ; 6153
                       18    A2         56  A0 00130          ADDW2   SCROLL_AMOUNT, 24(DISPTR)               ; 6154
                       0C    A2         56  A0 00134          ADDW2   SCROLL_AMOUNT, 12(DISPTR)               ; 6155
                             50      44 A2  D0 00138          MOVL    68(DISPTR), R0                          ; 6157
                             53         50  D1 0013C          CMPL    R0, CENTER_LINE                         ;
                                        03  15 0013F          BLEQ    12$                                     ;
                             50         53  D0 00141          MOVL    CENTER_LINE, R0                         ;
                             51      40 A2  D0 00144 12$:     MOVL    64(DISPTR), R1                          ;
                             50         51  D1 00148          CMPL    R1, R0                                  ;
                                        03  18 0014B          BGEQ    13$                                     ;
                             51         50  D0 0014D          MOVL    R0, R1                                  ;
                       38    A2         51  D0 00150 13$:     MOVL    R1, 56(DISPTR)                          ; 6156
          00000000G    00    38    A2   01  C1 00154          ADDL3   #1, 56(DISPTR), DBG$SRC_NEXT_LNUM       ; 6163
                       3C    A2         01  CE 0015D          MNEGL   #1, 60(DISPTR)                          ; 6171
                             59         57  D1 00161          CMPL    R7, R9                                  ; 6172
                                        07  12 00164          BNEQ    14$                                     ;
                             04         6A  E8 00166          BLBS    MARKLINE_DISABLE_FLAG, 14$              ;
                       3C    A2         53  D0 00169          MOVL    CENTER_LINE, 60(DISPTR)                 ; 6174
                                        6A  D4 0016D 14$:     CLRL    MARKLINE_DISABLE_FLAG                   ; 6176
                       18    BC         01  D0 0016F          MOVL    #1, @ALL_DONE_FLAG                      ; 6183
                                        04  00173          RET                                             ; 6018
                                 18    BC  D4 00174 15$:     CLRL    @ALL_DONE_FLAG                          ; 6192
FC    AA                 68         01     11  EF 00177          EXTZV   #17, #1, (R8), INVSCR_FLAG      ; 6199
                             40      FC AA  E8 0017D          BLBS    INVSCR_FLAG, 16$                        ; 6200
                                     1E A2  B5 00181          TSTW    30(DISPTR)                              ;
```

```
                                                3B  13 00184          BEQL    16$                                             ;
                                1718    CA      0C  A2 32 00186          CVTWL   12(DISPTR), SAVED_SCROLL                       ; 6203
                                1728    CA      18  A2 3C 0018C          MOVZWL  24(DISPTR), TOP_OFFSET                      ; 6204
                                                1728    CA  D7 00192          DECL    TOP_OFFSET                              ;
                                        50      1E  A2 3C 00196          MOVZWL  30(DISPTR), R0                              ; 6205
                                        51      18  A2 3C 0019A          MOVZWL  24(DISPTR), R1                              ;
                        C4      AA              50      51 C3 0019E          SUBL3   R1, R0, BOT_OFFSET                      ;
                                        54      20  A2 D0 001A3          MOVL    32(DISPTR), DLEPTR                          ; 6206
                                1720    CA      0A  A4 3C 001A7          MOVZWL  10(DLEPTR), TOP_REF_FILEID                  ; 6207
                                1730    CA      0C  A4 D0 001AD          MOVL    12(DLEPTR), TOP_REF_RECNUM                  ; 6208
                                        54      24  A2 D0 001B3          MOVL    36(DISPTR), DLEPTR                          ; 6209
                                  C8    AA      0A  A4 3C 001B7          MOVZWL  10(DLEPTR), BOT_REF_FILEID                  ; 6210
                                  CC    AA      0C  A4 D0 001BC          MOVL    12(DLEPTR), BOT_REF_RECNUM                  ; 6211
                                                52  DD 001C1 16$:        PUSHL   DISPTR                                      ; 6217
                                D89C    CF          01 FB 001C3          CALLS   #1, DBG$SCR_EMPTY_DISPLAY                   ;
                                  34    A2      14  AC D1 001C8          CMPL    MODRSTPTR, 52(DISPTR)                       ; 6225
                                                21  13 001CD          BEQL    17$                                             ;
                                  34    A2      14  AC D0 001CF          MOVL    MODRSTPTR, 52(DISPTR)                       ; 6228
                                  02    A8          02 88 001D4          BISB2   #2, 2(R8)                                   ; 6229
                                  3C    A2          01 CE 001D8          MNEGL   #1, 60(DISPTR)                              ; 6230
                                                44  A2 9F 001DC          PUSHAB  68(DISPTR)                                  ; 6232
                                                40  A2 9F 001DF          PUSHAB  64(DISPTR)                                  ;
                                                14  AC DD 001E2          PUSHL   MODRSTPTR                                   ;
                            00000000G   00          03 FB 001E5          CALLS   #3, DBG$SRC_LNUM_RANGE                      ;
                                  FC    AA          01 D0 001EC          MOVL    #1, INVSCR_FLAG                             ; 6233
                                  02    A8          0C 8A 001F0 17$:     BICB2   #12, 2(R8)                                  ; 6242
                        50              59          57 C3 001F4          SUBL3   R7, R9, R0                                  ; 6243
                                        50          02 C6 001F8          DIVL2   #2, R0                                      ;
                        53              50          57 C1 001FB          ADDL3   R7, R0, CENTER_LINE                         ;
                                        50      12  A2 3C 001FF          MOVZWL  18(DISPTR), R0                              ; 6244
                                        50      1C  A2 B1 00203          CMPW    28(DISPTR), R0                              ;
                                                04  1B 00207          BLEQU   18$                                             ;
                                        50      1C  A2 3C 00209          MOVZWL  28(DISPTR), R0                              ;
                                        56      0C  AC D0 0020D 18$:     MOVL    NEW_LOW, R6                                 ; 6245
                                        54      FF  A0 9E 00211          MOVAB   -1(R0), R4                                  ;
                                        54          02 C6 00215          DIVL2   #2, R4                                      ;
                        66              53          54 C3 00218          SUBL3   R4, CENTER_LINE, (R6)                       ;
                                        51      10  AC D0 0021C          MOVL    NEW_HIGH, R1                                ; 6246
                        54              50          02 C7 00220          DIVL3   #2, SIZE, R4                                ;
                        61              54          53 C1 00224          ADDL3   CENTER_LINE, R4, (R1)                       ;
                                  40    A2          66 D1 00228          CMPL    (R6), 64(DISPTR)                            ; 6247
                                                15  18 0022C          BGEQ    19$                                             ;
                                        66      40  A2 D0 0022F          MOVL    64(DISPTR), (R6)                            ; 6250
                        54              50          01 78 00232          ASHL    #1, SIZE, R4                                ; 6251
                                                54  D7 00236          DECL    R4                                              ;
                                        54          02 C6 00238          DIVL2   #2, R4                                      ;
                        61              54          66 C1 0023B          ADDL3   (R6), R4, (R1)                              ;
                                  02    A8          04 88 0023F          BISB2   #4, 2(R8)                                   ; 6252
                                  44    A2          61 D1 00243 19$:     CMPL    (R1), 68(DISPTR)                            ; 6255
                                                23  15 00247          BLEQ    21$                                             ;
                                        61      44  A2 D0 00249          MOVL    68(DISPTR), (R1)                            ; 6258
                                        50          02 C4 0024D          MULL2   #2, R0                                      ; 6260
                                                50  D7 00250          DECL    R0                                              ;
                                        50          02 C6 00252          DIVL2   #2, R0                                      ;
                        51              61          50 C3 00255          SUBL3   R0, (R1), R1                                ;
                                        50      40  A2 D0 00259          MOVL    64(DISPTR), R0                              ;
                                        51          50 D1 0025D          CMPL    R0, R1                                      ;
```

```
                                03  18 00260          BGEQ    20$                                     ;
                             50 51  D0 00262          MOVL    R1, R0                                  ;
                             66 50  D0 00265 20$:     MOVL    R0, (R6)                                ; 6259
                          02 A8 08  88 00268          BISB2   #8, 2(R8)                               ; 6261
                             50 44  A2 D0 0026C 21$:  MOVL    68(DISPTR), R0                          ; 6265
                             53 50  D1 00270          CMPL    R0, CENTER_LINE                         ;
                                03  15 00273          BLEQ    22$                                     ;
                             50 53  D0 00275          MOVL    CENTER_LINE, R0                         ;
                             51 40  A2 D0 00278 22$:  MOVL    64(DISPTR), R1                          ;
                             50 51  D1 0027C          CMPL    R1, R0                                  ;
                                03  18 0027F          BGEQ    23$                                     ;
                             51 50  D0 00281          MOVL    R0, R1                                  ;
                          38 A2 51  D0 00284 23$:     MOVL    R1, 56(DISPTR)                          ; 6264
                             59 57  D1 00288          CMPL    R7, R9                                  ; 6272
                                07  12 0028B          BNEQ    24$                                     ;
                             04 6A  E8 0028D          BLBS    MARKLINE_DISABLE_FLAG, 24$              ;
                          3C A2 53  D0 00290          MOVL    CENTER_LINE, 60(DISPTR)                 ; 6274
                             6A  D4 00294 24$:        CLRL    MARKLINE_DISABLE_FLAG                   ; 6276
                                 04 00296            RET                                             ; 6279

; Routine Size:  663 bytes,    Routine Base:  DBG$CODE + 33FF
```

```
;  6194          6280  1 GLOBAL ROUTINE DBG$SCR_SOURCE_END: NOVALUE =
;  6195          6281  1 !
;  6196          6282  1 ! FUNCTION
;  6197          6283  1 !       This routine is called at the end of a source line output operation
;  6198          6284  1 !       to a screen display.  It is called from DBG$SRC_TYPE_LNUM_SOURCE if
;  6199          6285  1 !       source lines are being directed to a screen display.  The primary
;  6200          6286  1 !       purpose of this routine is to determine where the screen window of
;  6201          6287  1 !       the source output display should be positioned within the display
;  6202          6288  1 !       text.  This computation, plus the computation of a scrolling count
;  6203          6289  1 !       for the display, is accomplished by doing some bookkeeping based on
;  6204          6290  1 !       a number of OWN variables set up in routines DBG$SCR_SOURCE_BEGIN,
;  6205          6291  1 !       DBG$SCR_SCROLL_SOURCE_DOWN, and DBG$SCR_SCROLL_SOURCE_UP.
;  6206          6292  1 !
;  6207          6293  1 !       The fundamental idea behind the algorithm is that SOURCE_BEGIN has
;  6208          6294  1 !       marked the top and bottom lines of the previous display contents
;  6209          6295  1 !       as "reference points" with known source file File IDs and record
;  6210          6296  1 !       numbers.  SOURCE_END then searches the new display contents for a
;  6211          6297  1 !       line with the same File ID and record number as one of these two
;  6212          6298  1 !       reference points.  If such a line is found, the corresponding
;  6213          6299  1 !       scrolling count (if DROW is fixed by the user's TYPE or EXAM/SOURCE
;  6214          6300  1 !       command) or the corresponding DROW value (if the scrolling count is
;  6215          6301  1 !       fixed by the user's SCROLL command) can be computed for the display.
;  6216          6302  1 !       The whole purpose of all these computations is thus to permit proper
;  6217          6303  1 !       vertical scrolling of source displays on the terminal screen.
;  6218          6304  1 !
;  6219          6305  1 ! INPUTS
;  6220          6306  1 !       NONE
;  6221          6307  1 !
;  6222          6308  1 ! OUTPUTS
;  6223          6309  1 !       NONE
;  6224          6310  1 !
;  6225          6311  1
;  6226          6312  2     BEGIN
;  6227          6313  2
;  6228          6314  2     LOCAL
;  6229          6315  2         DISPTR: REF DBG$DISP_ENTRY,      ! Pointer to Screen Display Entry
;  6230          6316  2         DLEPTR: REF DBG$DLINE_ENTRY,     ! Pointer to Display Line Entry
;  6231          6317  2         DROW,                            ! Display row location of window top
;  6232          6318  2         NEW_TOP_OFFSET,                  ! Offset from top of display to the
;  6233          6319  2                                          !       source reference line
;  6234          6320  2         REF_FLAG,                        ! Flag set if a reference line is found
;  6235          6321  2         SCROLL_DOWN;                     ! Flag set if we should scroll down
;  6236          6322  2
;  6237          6323  2
;  6238          6324  2
;  6239          6325  2     ! Set the next line number (as used by the parameter-less TYPE and SEARCH
;  6240          6326  2     ! commands) to be the line after the center line for the current source
;  6241          6327  2     ! display.
;  6242          6328  2     !
;  6243          6329  2     DISPTR = .DBG$GL_SCREEN_SOURCE;
;  6244          6330  2     DBG$SRC_NEXT_LNUM = .DISPTR[DBG$L_DISP_CENTER] + 1;
;  6245          6331  2
;  6246          6332  2
;  6247          6333  2     ! Search the new list of Display Line Entries to see if we can find the
;  6248          6334  2     ! reference line that used to be at the top or the bottom of the previous
;  6249          6335  2     ! display contents.  If we can, note whether this calls for scrolling up
;  6250          6336  2     ! or down.
```

```
;  6251          6337  2      !
;  6252          6338  2      NEW_TOP_OFFSET = 0;
;  6253          6339  2      REF_FLAG = FALSE;
;  6254          6340  2      DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  6255          6341  2      WHILE .DLEPTR NEQ DISPTR[DBG$L_DISP_START_LINE_PTR] DO
;  6256          6342  3          BEGIN
;  6257          6343  3          IF (.DLEPTR[DBG$L_DLINE_RECNUM] EQL .TOP_REF_RECNUM) AND
;  6258          6344  4             (.DLEPTR[DBG$W_DLINE_FILEID] EQL .TOP_REF_FILEID)
;  6259          6345  3          THEN
;  6260          6346  4              BEGIN
;  6261          6347  4              REF_FLAG = TRUE;
;  6262          6348  4              SCROLL_DOWN = FALSE;
;  6263          6349  4              EXITLOOP;
;  6264          6350  3              END;
;  6265          6351  3
;  6266          6352  3          IF (.DLEPTR[DBG$L_DLINE_RECNUM] EQL .BOT_REF_RECNUM) AND
;  6267          6353  4             (.DLEPTR[DBG$W_DLINE_FILEID] EQL .BOT_REF_FILEID)
;  6268          6354  3          THEN
;  6269          6355  4              BEGIN
;  6270          6356  4              REF_FLAG = TRUE;
;  6271          6357  4              SCROLL_DOWN = TRUE;
;  6272          6358  4              EXITLOOP;
;  6273          6359  3              END;
;  6274          6360  3
;  6275          6361  3          NEW_TOP_OFFSET = .NEW_TOP_OFFSET + 1;
;  6276          6362  3          DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  6277          6363  2          END;
;  6278          6364  2
;  6279          6365  2
;  6280          6366  2      ! If we did not find either reference line, we invalidate the scrolling
;  6281          6367  2      ! count.
;  6282          6368  2      !
;  6283          6369  2      IF NOT .REF_FLAG THEN INVSCR_FLAG = TRUE;
;  6284          6370  2
;  6285          6371  2
;  6286          6372  2      ! If we found a reference line and we are scrolling by a fixed amount
;  6287          6373  2      ! specified by the user on a SCROLL/DOWN or SCROLL/UP command, we compute
;  6288          6374  2      ! the new DROW location.  Here we use SCROLL_AMOUNT as set in routine
;  6289          6375  2      ! DBG$SCR_SCROLL_SOURCE_DOWN or DBG$SCR_SCROLL_SOURCE_UP to compute DROW.
;  6290          6376  2      ! DROW is the adjusted if it hits the top or bottom of the source display.
;  6291          6377  2      !
;  6292          6378  2      DROW = .DISPTR[DBG$W_DISP_DROW];
;  6293          6379  2      IF .FIXED_SCROLL_AMOONT AND .REF_FLAG
;  6294          6380  2      THEN
;  6295          6381  3          BEGIN
;  6296          6382  3
;  6297          6383  3
;  6298          6384  3          ! If we are scrolling down and the scrolling amount is fixed by the
;  6299          6385  3          ! user's SCROLL/DOWN command, we compute the resulting DROW value here.
;  6300          6386  3          !
;  6301          6387  3          IF .SCROLL_DOWN
;  6302          6388  3          THEN
;  6303          6389  3              DROW = .NEW_TOP_OFFSET + (.SCROLL_AMOUNT - .BOT_OFFSET) + 1
;  6304          6390  3
;  6305          6391  3
;  6306          6392  3          ! If we are scrolling up and the scrolling amount is fixed by the
;  6307          6393  3          ! user's SCROLL/UP command, we compute the resulting DROW value here.
```

```
; 6308          6394  3          !
; 6309          6395  3          ELSE
; 6310          6396  3              DROW = .NEW_TOP_OFFSET + .SCROLL_AMOUNT + .TOP_OFFSET + 1;
; 6311          6397  3
; 6312          6398  3          END                                 ! End of fixed-scrolling-amount case
; 6313          6399  3
; 6314          6400  3
; 6315          6401  3      ! If we found a reference line but the scrolling amount is not fixed, then
; 6316          6402  3      ! the center line of the window was specified by the user on a TYPE or
; 6317          6403  3      ! EXAMINE/SOURCE command.  Here we compute the corresponding scrolling
; 6318          6404  3      ! amount for the window.  DROW has already been set in DBG$SCR_SOURCE_LINE
; 6319          6405  3      ! in this case.
; 6320          6406  3      !
; 6321          6407  2      ELSE IF .REF_FLAG
; 6322          6408  2      THEN
; 6323          6409  3          BEGIN
; 6324          6410  3
; 6325          6411  3
; 6326          6412  3          ! If we are scrolling down and the center line of the window is fixed
; 6327          6413  3          ! by the source line number specified on the TYPE or EXAM/SOURCE com-
; 6328          6414  3          ! mand, we compute the resulting scrolling amount here.
; 6329          6415  3          !
; 6330          6416  3          IF .SCROLL_DOWN
; 6331          6417  3          THEN
; 6332          6418  3              SCROLL_AMOUNT = .BOT_OFFSET + .DROW - 1 - .NEW_TOP_OFFSET
; 6333          6419  3
; 6334          6420  3
; 6335          6421  3          ! If we are scrolling up and the center line of the window is fixed by
; 6336          6422  3          ! the source line number specified on the TYPE or EXAM/SOURCE command,
; 6337          6423  3          ! we compute the resulting scrolling amount here.
; 6338          6424  3          !
; 6339          6425  3          ELSE
; 6340          6426  3              SCROLL_AMOUNT = -(.NEW_TOP_OFFSET + .TOP_OFFSET - .DROW+ 1);
; 6341          6427  3
; 6342          6428  3          END                                 ! End of fixed DROW case
; 6343          6429  3
; 6344          6430  3
; 6345          6431  3      ! If no reference line was found but we are scrolling to the bottom of the
; 6346          6432  3      ! display, simply set DROW to point to the last line of the display.  (This
; 6347          6433  3      ! gets adjusted for the window size in the code immediately below.)
; 6348          6434  3      !
; 6349          6435  2      ELSE IF .FIXED_SCROLL_AMOUNT AND .SCROLL_TO_BOTTOM
; 6350          6436  2      THEN
; 6351          6437  2          DROW = .DISPTR[DBG$W_DISP_LINECNT];
; 6352          6438  2
; 6353          6439  2
; 6354          6440  2      ! Then adjust the DROW value and the scrolling amount if we are below
; 6355          6441  2      ! the bottom of the source file being scrolled.
; 6356          6442  2      !
; 6357          6443  2      IF .DROW - 1 GTR .DISPTR[DBG$W_DISP_LINECNT] - .DISPTR[DBG$W_DISP_RLEN]
; 6358          6444  2      THEN
; 6359          6445  3          BEGIN
; 6360          6446  3          SCROLL_AMOUNT = .SCROLL_AMOUNT + .DISPTR[DBG$W_DISP_LINECNT]
; 6361          6447  3                                         - .DISPTR[DBG$W_DISP_RLEN] - .DROW + 1;
; 6362          6448  3          DROW = .DISPTR[DBG$W_DISP_LINECNT] - .DISPTR[DBG$W_DISP_RLEN] + 1;
; 6363          6449  2          END;
; 6364          6450  2
```

```
;  6365          6451  2
;  6366          6452  2      ! Next adjust the DROW value and the scrolling amount if we are above
;  6367          6453  2      ! the top of the source file being scrolled.  DROW could be zero or
;  6368          6454  2      ! negative due to either of the above DROW computations; here we make
;  6369          6455  2      ! sure it does not get smaller that 1.
;  6370          6456  2      !
;  6371          6457  2      IF .DROW LSS 1
;  6372          6458  2      THEN
;  6373          6459  3          BEGIN
;  6374          6460  3          SCROLL_AMOUNT = .SCROLL_AMOUNT - .DROW + 1;
;  6375          6461  3          DROW = 1;
;  6376          6462  2          END;
;  6377          6463  2
;  6378          6464  2
;  6379          6465  2      ! Set the final DROW value into the Screen Display Entry.
;  6380          6466  2      !
;  6381          6467  2      DISPTR[DBG$W_DISP_DROW] = .DROW;
;  6382          6468  2
;  6383          6469  2
;  6384          6470  2      ! Given the new DROW (display row) value, set up the display's window
;  6385          6471  2      ! start pointer.
;  6386          6472  2      !
;  6387          6473  2      DLEPTR = DISPTR[DBG$L_DISP_START_LINE_PTR];
;  6388          6474  2      INCR I FROM 1 TO .DISPTR[DBG$W_DISP_DROW] DO
;  6389          6475  2          DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  6390          6476  2
;  6391          6477  2      DISPTR[DBG$L_DISP_WINDOW_PTR] = .DLEPTR;
;  6392          6478  2
;  6393          6479  2
;  6394          6480  2      ! Compute the final scrolling count, mark its validity, and return.
;  6395          6481  2      !
;  6396          6482  2      DISPTR[DBG$W_DISP_SCROLL] = .SAVED_SCROLL + .SCROLL_AMOUNT;
;  6397          6483  2      DISPTR[DBG$V_DISP_INVSCR] = .INVSCR_FLAG;
;  6398          6484  2      RETURN;
;  6399          6485  2
;  6400          6486  1      END;


                                                007C 00000          .ENTRY  DBG$SCR_SOURCE_END, Save R2,R3,R4,R5,R6      ; 6280
                                            56 00000000'  EF  9E 00002      MOVAB   SCROLL_AMOUNT, R6                            ;
                                            50 00000000'  EF  D0 00009      MOVL    DBG$GL_SCREEN_SOURCE, DISPTR         ; 6329
                00000000G  00       38      A0              0' C1 00010     ADDL3   #1, 56(DISPTR), DBG$SRC_NEXT_LNUM    ; 6330
                                                51  D4 00019      CLRL    NEW_TOP_OFFSET                       ; 6338
                                                55  D4 0001B      CLRL    REF_FLAG                             ; 6339
                                            53      20      A0  D0 0001D      MOVL    32(DISPTR), DLEPTR                   ; 6340
                                            52      20      A0  9E 00021 1$:  MOVAB   32(DISPTR), R2                       ; 6341
                                            52              53  D1 00025      CMPL    DLEPTR, R2                           ;
                                                            38  13 00028      BEQL    4$                                   ;
                                        10  A6      0C      A3  D1 0002A      CMPL    12(DLEPTR), TOP_REF_RECNUM           ; 6343
                                                            10  12 0002F      BNEQ    2$                                   ;
        0C  A6          0A  A3          10                  00  ED 00031      CMPZV   #0, #16, 10(DLEPTR), TOP_REF_FILEID  ; 6344
                                                            07  12 00038      BNEQ    2$                                   ;
                                            55              01  D0 0003A      MOVL    #1, REF_FLAG                         ; 6347
                                                            54  D4 0003D      CLRL    SCROLL_DOWN                          ; 6348
```

```
                                                21  11 0003F                BRB     4$                                      ; 6346
                                        E8AC    C6      0C      A3  D1 00041 2$:      CMPL    12(DLEPTR), BOT_REF_RECNUM              ; 6352
                                                12  12 00047                BNEQ    3$
    E8A8    C6          0A      A3              10              00  ED 00049        CMPZV   #0, #16, 10(DLEPTR), BOT_REF_FILEID      ; 6353
                                                08  12 00051                BNEQ    3$
                                            55      01  D0 00053            MOVL    #1, REF_FLAG                            ; 6356
                                            54      01  D0 00056            MOVL    #1, SCROLL_DOWN                         ; 6357
                                                07  11 00059                BRB     4$                                      ; 6355
                                                51  D6 0005B 3$:           INCL    NEW_TOP_OFFSET                          ; 6361
                                            53      63  D0 0005D            MOVL    (DLEPTR), DLEPTR                        ; 6362
                                                BF  11 00060                BRB     1$                                      ; 6341
                                            05      55  E8 00062 4$:       BLBS    REF_FLAG, 5$                            ; 6369
                                        E8DC    C6      01  D0 00065        MOVL    #1, INVSCR_FLAG
                                            52      18      A0  3C 0006A 5$:  MOVZWL  24(DISPTR), DROW                        ; 6378
                                            21      E8D4    C6  E9 0006E    BLBC    FIXED_SCROLL_AMOUNT, 7$                 ; 6379
                                            41          55  E9 00073        BLBC    REF_FLAG, 9$
                                            0D          54  E9 00076        BLBC    SCROLL_DOWN, 6$                         ; 6387
                                54          66      E8A4    C6  C3 00079    SUBL3   BOT_OFFSET, SCROLL_AMOUNT, R4           ; 6389
                                            52      01  A441    9E 0007F    MOVAB   1(R4)[NEW_TOP_OFFSET], DROW
                                                3E  11 00084                BRB     10$
                                54          51          66  C1 00086 6$:  ADDL3   SCROLL_AMOUNT, NEW_TOP_OFFSET, R4       ; 6396
                                            54      08      A6  C0 0008A    ADDL2   TOP_OFFSET, R4
                                            52      01      A4  9E 0008E    MOVAB   1(R4), DROW
                                                30  11 00092                BRB     10$                                     ; 6379
                                            20          55  E9 00094 7$:    BLBC    REF_FLAG, 9$                            ; 6407
                                            0F          54  E9 00097        BLBC    SCROLL_DOWN, 8$                         ; 6416
                                54          52      E8A4    C6  C1 0009A    ADDL3   BOT_OFFSET, DROW, R4                    ; 6418
                                            54          51  C2 000A0        SUBL2   NEW_TOP_OFFSET, R4
                                            66          FF  A4  9E 000A3    MOVAB   -1(R4), SCROLL_AMOUNT
                                                1B  11 000A7                BRB     10$
                                            51      08      A6  C0 000A9 8$:  ADDL2   TOP_OFFSET, R1                          ; 6426
                                            51          52  C2 000AD        SUBL2   DROW, R1
                                                51  D6 000B0                INCL    R1
                                            66          51  CE 000B2        MNEGL   R1, SCROLL_AMOUNT
                                                0D  11 000B5                BRB     10$                                     ; 6407
                                            08      E8D4    C6  E9 000B7 9$:  BLBC    FIXED_SCROLL_AMOUNT, 10$                ; 6435
                                            04      04      A6  E9 000BC    BLBC    SCROLL_TO_BOTTOM, 10$
                                            52      1E      A0  3C 000C0    MOVZWL  30(DISPTR), DROW                        ; 6437
                                            51      FF      A2  9E 000C4 10$:  MOVAB   -1(R2), R1                             ; 6443
                                            54      1E      A0  3C 000C8    MOVZWL  30(DISPTR), R4
                                            55      12      A0  3C 000CC    MOVZWL  18(DISPTR), R5
                                            54          55  C2 000D0        SUBL2   R5, R4
                                            54          51  D1 000D3        CMPL    R1, R4
                                                19  15 000D6                BLEQ    11$
                                            51      1E      A0  3C 000D8    MOVZWL  30(DISPTR), R1                          ; 6446
                                            51          66  C0 000DC        ADDL2   SCROLL_AMOUNT, R1
                                            55      12      A0  3C 000DF    MOVZWL  18(DISPTR), R5                          ; 6447
                                            51          55  C2 000E3        SUBL2   R5, R1
                                            51          52  C2 000E6        SUBL2   DROW, R1
                                            66      01      A1  9E 000E9    MOVAB   1(R1), SCROLL_AMOUNT
                                            52      01      A4  9E 000ED    MOVAB   1(R4), DROW                             ; 6448
                                                52  D5 000F1 11$:  TSTL    DROW                                            ; 6457
                                                0B  14 000F3                BGTR    12$
                51              66              52  C3 000F5            SUBL3   DROW, SCROLL_AMOUNT, R1                 ; 6460
                                66      01      A1  9E 000F9            MOVAB   1(R1), SCROLL_AMOUNT
                                            52      01  D0 000FD        MOVL    #1, DROW                                ; 6461
                        18      A0              52  B0 00100 12$:      MOVW    DROW, 24(DISPTR)                        ; 6467
```

```
                                53      20    A0  9E 00104          MOVAB   32(DISPTR), DLEPTR                          ; 6473
                                52      18    A0  3C 00108          MOVZWL  24(DISPTR), R2                           ; 6474
                                              51  D4 0010C          CLRL    I                                        ;
                                              03  11 0010E          BRB     14$                                      ;
                                53            63  D0 00110 13$:     MOVL    (DLEPTR), DLEPTR                         ; 6475
                        F9      51            52  F3 00113 14$:     AOBLEQ  R2, I, 13$                               ;
                    28  A0      53            53  D0 00117          MOVL    DLEPTR, 40(DISPTR)                       ; 6477
                0C  A0  F8  A6                66  A1 0011B          ADDW3   SCROLL_AMOUNT, SAVED_SCROLL, 12(DISPTR)  ; 6482
        0A  A0          01      01    E8DC    C6  F0 00121          INSV    INVSCR_FLAG, #1, #1, 10(DISPTR)          ; 6483
                                              04  00129          RET                                              ; 6486

; Routine Size:  298 bytes,    Routine Base:  DBG$CODE + 3696
```

```
;  6402          6487  1 GLOBAL ROUTINE DBG$SCR_SOURCE_LINE(LINENUM, FILEID, RECNUM,
;  6403          6488  1                                    BUFLEN, BUFPTR): NOVALUE =
;  6404          6489  1 !
;  6405          6490  1 ! FUNCTION
;  6406          6491  1 !       This routine inserts a source line into a SOURCE screen display.
;  6407          6492  1 !       It accepts the line number, source file location, and text string
;  6408          6493  1 !       of the source line as input and it inserts that text and other
;  6409          6494  1 !       information into the current source display as specified by the
;  6410          6495  1 !       global pointer DBG$GL_SCREEN_SOURCE.
;  6411          6496  1 !
;  6412          6497  1 ! INPUTS
;  6413          6498  1 !       LINENUM - The line number of the passed-in source line.  This is the
;  6414          6499  1 !                 line number which will be displayed to the left of the line
;  6415          6500  1 !                 on the screen display.
;  6416          6501  1 !
;  6417          6502  1 !       FILEID  - The source file File ID as understood by module DBGSOURCE.
;  6418          6503  1 !                 FILEID and RECNUM serve to uniquely identify this source
;  6419          6504  1 !                 line.
;  6420          6505  1 !
;  6421          6506  1 !       RECNUM  - The record number of the source line within the FILEID
;  6422          6507  1 !                 source file.  Again, the purpose of FILEID and RECNUM is
;  6423          6508  1 !                 to uniquely identify this source line.
;  6424          6509  1 !
;  6425          6510  1 !       BUFLEN  - The length of the source line text in characters.
;  6426          6511  1 !
;  6427          6512  1 !       BUFPTR  - The address of the source line text.
;  6428          6513  1 !
;  6429          6514  1 ! OUTPUTS
;  6430          6515  1 !       NONE
;  6431          6516  1 !
;  6432          6517  1
;  6433          6518  2     BEGIN
;  6434          6519  2
;  6435          6520  2     MAP
;  6436          6521  2         BUFPTR: REF VECTOR[,BYTE];      ! Pointer to source line text
;  6437          6522  2
;  6438          6523  2     LOCAL
;  6439          6524  2         BLINK: REF DBG$DLINE_ENTRY,     ! Pointer to previous Display Line Entry
;  6440          6525  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to SOURCE Screen Display Entry
;  6441          6526  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Screen Display Line Entry
;  6442          6527  2         FLINK: REF DBG$DLINE_ENTRY,     ! Pointer to next Display Line Entry
;  6443          6528  2         TEMPBUF: VECTOR[256,BYTE],      ! Temporary buffer for de-tabbed text
;  6444          6529  2         TLEN,                           ! Length of de-tabbed text
;  6445          6530  2         TEXT_PTR: REF VECTOR[,BYTE];    ! Pointer to text in Display Line Entry
;  6446          6531  2
;  6447          6532  2
;  6448          6533  2
;  6449          6534  2     ! Expand all tabs to blanks.  This is necessary in order to make subsequent
;  6450          6535  2     ! length computations and left scrolling work correctly.  The expanded text
;  6451          6536  2     ! is copied into the TEMPBUF temporary buffer.
;  6452          6537  2     !
;  6453          6538  2     TLEN = 0;
;  6454          6539  2     INCR I FROM 0 TO .BUFLEN - 1 DO
;  6455          6540  3         BEGIN
;  6456          6541  3
;  6457          6542  3
;  6458          6543  3         ! If the current character is a tab character, expand it to blanks.
```

```
;  6459          6544  3          !
;  6460          6545  3          IF .BUFPTR[.I] EQL TAB_CHAR
;  6461          6546  3          THEN
;  6462          6547  4              BEGIN
;  6463          6548  4              INCR J FROM 1 TO 8 - (.TLEN MOD 8) DO
;  6464          6549  5                  BEGIN
;  6465          6550  5                  TEMPBUF[.TLEN] = ' ';
;  6466          6551  5                  TLEN = .TLEN + 1;
;  6467          6552  5                  IF .TLEN GEQ 255 THEN EXITLOOP;
;  6468          6553  4                  END;
;  6469          6554  4
;  6470          6555  4              END
;  6471          6556  4
;  6472          6557  4
;  6473          6558  4          ! If the current character is not a tab, just copy it as is.
;  6474          6559  4          !
;  6475          6560  3          ELSE
;  6476          6561  4              BEGIN
;  6477          6562  4              TEMPBUF[.TLEN] = .BUFPTR[.I];
;  6478          6563  4              TLEN = .TLEN + 1;
;  6479          6564  3              END;
;  6480          6565  3
;  6481          6566  3          IF .TLEN GEQ 255 THEN EXITLOOP;
;  6482          6567  2          END;
;  6483          6568  2
;  6484          6569  2
;  6485          6570  2      ! Allocate and build a Screen Display Line Entry to hold the contents of
;  6486          6571  2      ! the present source line.
;  6487          6572  2      !
;  6488          6573  2      DISPTR = .DBG$GL_SCREEN_SOURCE;
;  6489          6574  2      DLEPTR = DBG$GET_MEMORY(DBG$K_DLINE_ENTSIZE2 + .TLEN/%UPVAL + 1);
;  6490          6575  2      DLEPTR[DBG$B_DLINE_REND] = .DISPTR[DBG$B_DISP_REND];
;  6491          6576  2      DLEPTR[DBG$V_DLINE_SOURCEFLG] = TRUE;
;  6492          6577  2      DLEPTR[DBG$W_DLINE_FILEID] = .FILEID;
;  6493          6578  2      DLEPTR[DBG$L_DLINE_RECNUM] = .RECNUM;
;  6494          6579  2      DLEPTR[DBG$L_DLINE_LINUM] = .LINENUM;
;  6495          6580  2
;  6496          6581  2
;  6497          6582  2      ! Copy over the text of the source line to the Display Line Entry.
;  6498          6583  2      !
;  6499          6584  2      TEXT_PTR = DLEPTR[DBG$A_DLINE_TEXT2];
;  6500          6585  2      TEXT_PTR[0] = .TLEN;
;  6501          6586  2      CH$MOVE(.TLEN, TEMPBUF, TEXT_PTR[1]);
;  6502          6587  2
;  6503          6588  2
;  6504          6589  2      ! Link this Display Line Entry onto the tail of the Display Line Entry list
;  6505          6590  2      ! for this display.
;  6506          6591  2      !
;  6507          6592  2      FLINK = DISPTR[DBG$L_DISP_START_LINE_PTR];
;  6508          6593  2      BLINK = .FLINK[DBG$L_DLINE_BLINK];
;  6509          6594  2      DLEPTR[DBG$L_DLINE_FLINK] = .FLINK;
;  6510          6595  2      DLEPTR[DBG$L_DLINE_BLINK] = .BLINK;
;  6511          6596  2      FLINK[DBG$L_DLINE_BLINK] = .DLEPTR;
;  6512          6597  2      BLINK[DBG$L_DLINE_FLINK] = .DLEPTR;
;  6513          6598  2      DISPTR[DBG$W_DISP_LINECNT] = .DISPTR[DBG$W_DISP_LINECNT] + 1;
;  6514          6599  2      IF .DISPTR[DBG$W_DISP_LINECNT] EQL 1
;  6515          6600  2      THEN
```

```
;  6516          6601  2          DISPTR[DBG$L_DISP_WINDOW_PTR] = .DLEPTR;
;  6517          6602  2
;  6518          6603  2
;  6519          6604  2      ! Finally adjust the DROW parameter and window pointer so that the central
;  6520          6605  2      ! line of the source display in fact winds up in the center of the center
;  6521          6606  2      ! of the display's screen window.
;  6522          6607  2      !
;  6523          6608  2      IF .LINENUM EQL .DISPTR[DBG$L_DISP_CENTER]
;  6524          6609  2      THEN
;  6525          6610  3          BEGIN
;  6526          6611  3          DISPTR[DBG$W_DISP_DROW] = MAX(1, .DISPTR[DBG$W_DISP_LINECNT] -
;  6527          6612  3                                           (.DISPTR[DBG$W_DISP_RLEN] - 1)/2);
;  6528          6613  3          INCR I FROM 1 TO
;  6529          6614  3                  .DISPTR[DBG$W_DISP_LINECNT] - .DISPTR[DBG$W_DISP_DROW] DO
;  6530          6615  3              DLEPTR = .DLEPTR[DBG$L_DLINE_BLINK];
;  6531          6616  3
;  6532          6617  3          DISPTR[DBG$L_DISP_WINDOW_PTR] = .DLEPTR;
;  6533          6618  2          END;
;  6534          6619  2
;  6535          6620  2
;  6536          6621  2      ! The source line has been output to the display.  Now return.
;  6537          6622  2      !
;  6538          6623  2      RETURN;
;  6539          6624  2
;  6540          6625  1      END;


                                                00FC 00000          .ENTRY  DBG$SCR_SOURCE_LINE, Save R2,R3,R4,R5,R6,R7 ; 6487
                                      5E      FF00    CE   9E 00002         MOVAB   -256(SP), SP                        ;
                                              52      D4   00007         CLRL    TLEN                                ; 6538
                                      50              01   CE   00009         MNEGL   #1, I                               ; 6545
                                              3B      11   0000C         BRB     6$                                  ;
                                      09        14    BC40 91   0000E 1$:      CMPB    @BUFPTR[I], #9                      ;
                                              25      12   00013         BNEQ    4$                                  ;
7E                    00                52      01   7A   00015         EMUL    #1, TLEN, #0, -(SP)                 ; 6548
51                    51                8E      08   7B   0001A         EDIV    #8, (SP)+, R1, R1                   ;
                      51                08      51   C3   0001F         SUBL3   R1, #8, R1                          ;
                                              53      D4   00023         CLRL    J                                   ;
                                              0D      11   00025         BRB     3$                                  ;
                                    824E              20   90   00027 2$:      MOVB    #32, TEMPBUF[SP]                    ; 6550
                          000000FF    8F              52   D1   0002B         CMPL    TLEN, #255                          ; 6552
                                              0C      18   00032         BGEQ    5$                                  ;
                          EF          53              51   F3   00034 3$:      AOBLEQ  R1, J, 2$                   ; 6548
                                              06      11   00038         BRB     5$                                  ; 6545
                                    824E        14    BC40 90   0003A 4$:      MOVB    @BUFPTR[I], TEMPBUF[SP]             ; 6562
                          000000FF    8F              52   D1   00040 5$:      CMPL    TLEN, #255                          ; 6566
                                              05      18   00047         BGEQ    7$                                  ;
                          C0          50        10    AC   F2   00049 6$:      AOBLSS  BUFLEN, I, 1$               ; 6539
                                      56  00000000'    EF   D0   0004E 7$:      MOVL    DBG$GL_SCREEN_SOURCE, DISPTR        ; 6573
                          50          52              04   C7   00055         DIVL3   #4, TLEN, R0                        ; 6574
                                              06      A0   9F   00059         PUSHAB  6(R0)                               ;
                              00000000G   00          01   FB   0005C         CALLS   #1, DBG$GET_MEMORY                  ;
                                      57              50   D0   00063         MOVL    R0, DLEPTR                          ;
                                08    A7        09    A6   90   00066         MOVB    9(DISPTR), 8(DLEPTR)                ; 6575
```

```
                09   A7              02  88 0006B          BISB2   #2, 9(DLEPTR)                          ; 6576
                0A   A7        08    AC  B0 0006F          MOVW    FILEID, 10(DLEPTR)                     ; 6577
                0C   A7        0C    AC  D0 00074          MOVL    RECNUM, 12(DLEPTR)                     ; 6578
                10   A7        04    AC  D0 00079          MOVL    LINENUM, 16(DLEPTR)                    ; 6579
                     50        14    A7  9E 0007E          MOVAB   20(R7), TEXT_PTR                       ; 6584
                     60              52  90 00082          MOVB    TLEN, (TEXT_PTR)                       ; 6585
          01    A0   6E              52  28 00085          MOVC3   TLEN, TEMPBUF, 1(TEXT_PTR)             ; 6586
                     50        20    A6  9E 0008A          MOVAB   32(R6), FLINK                          ; 6592
                     51        04    A0  D0 0008E          MOVL    4(FLINK), BLINK                        ; 6593
                     67              50  7D 00092          MOVQ    FLINK, (DLEPTR)                        ; 6594
                04   A0              57  D0 00095          MOVL    DLEPTR, 4(FLINK)                       ; 6596
                     61              57  D0 00099          MOVL    DLEPTR, (BLINK)                        ; 6597
                           1E    A6  B6 0009C          INCW    30(DISPTR)                             ; 6598
                     01    1E    A6  B1 0009F          CMPW    30(DISPTR), #1                         ; 6599
                                   04  12 000A3          BNEQ    8$                                     ;
                28   A6              57  D0 000A5          MOVL    DLEPTR, 40(DISPTR)                     ; 6601
                38   A6        04    AC  D1 000A9 8$:      CMPL    LINENUM, 56(DISPTR)                    ; 6608
                                   35  12 000AE          BNEQ    12$                                    ;
                     50        12    A6  3C 000B0          MOVZWL  18(DISPTR), R0                         ; 6612
                                   50  D7 000B4          DECL    R0                                     ;
                     50              02  C6 000B6          DIVL2   #2, R0                                 ;
                     51        1E    A6  3C 000B9          MOVZWL  30(DISPTR), R1                         ;
          50         51              50  C3 000BD          SUBL3   R0, R1, R0                             ;
                                   03  14 000C1          BGTR    9$                                     ;
                     50              01  D0 000C3          MOVL    #1, R0                                 ; 6611
                18   A6              50  B0 000C6 9$:      MOVW    R0, 24(DISPTR)                         ;
                     50        1E    A6  3C 000CA          MOVZWL  30(DISPTR), R0                         ; 6614
                     51        18    A6  3C 000CE          MOVZWL  24(DISPTR), R1                         ;
                     50              51  C2 000D2          SUBL2   R1, R0                                 ;
                                   51  D4 000D5          CLRL    I                                      ; 6613
                                   04  11 000D7          BRB     11$                                    ;
                     57        04    A7  D0 000D9 10$:     MOVL    4(DLEPTR), DLEPTR                      ; 6615
          F8         51              50  F3 000DD 11$:     AOBLEQ  R0, I, 10$                             ;
                28   A6              57  D0 000E1          MOVL    DLEPTR, 40(DISPTR)                     ; 6617
                                   04  000E5 12$:      RET                                            ; 6625

; Routine Size:  230 bytes,    Routine Base:  DBG$CODE + 37C0
```

```
;  6542          6626  1 ROUTINE DBG$SCR_UPDATE_PASTEBOARD: NOVALUE =
;  6543          6627  1 !
;  6544          6628  1 ! FUNCTION
;  6545          6629  1 !       This routine updates the pasteboard to take into account the changes
;  6546          6630  1 !       made to the constituent displays since the last time the pasteboard
;  6547          6631  1 !       was updated.  It goes through the list of pasted displays to determine
;  6548          6632  1 !       which lines of which displays should be visible on the pasteboard, and
;  6549          6633  1 !       it updates the pasteboard line table accordingly.
;  6550          6634  1 !
;  6551          6635  1 ! INPUTS
;  6552          6636  1 !       NONE
;  6553          6637  1 !
;  6554          6638  1 ! OUTPUTS
;  6555          6639  1 !       NONE
;  6556          6640  1 !
;  6557          6641  1
;  6558          6642  2     BEGIN
;  6559          6643  2
;  6560          6644  2     LOCAL
;  6561          6645  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to current Display Entry
;  6562          6646  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Display Line Entry
;  6563          6647  2         IROW,                           ! Temporary row index into pasteboard
;  6564          6648  2         NEXTDLE,                        ! Pointer to next Display Line Entry
;  6565          6649  2         RBEG,                           ! Beginning row location on pasteboard
;  6566          6650  2                                         !       of the current display
;  6567          6651  2         RLEN,                           ! Row length (height) of current display
;  6568          6652  2         SCROLL_AMOUNT,                  ! Scrolling amount for current display
;  6569          6653  2         TEMPTR: REF DBG$DISP_ENTRY;     ! Temporary Display Entry pointer
;  6570          6654  2
;  6571          6655  2
;  6572          6656  2
;  6573          6657  2     ! Initialize the pasteboard to contain nothing but user scrolling region.
;  6574          6658  2     !
;  6575          6659  2     INCR I FROM 0 TO DBG$K_PASTE_SIZE - 1 DO
;  6576          6660  2         PASTEBOARD[.I, DBG$B_PASTE_KIND] = DBG$K_PASTE_NULL;
;  6577          6661  2
;  6578          6662  2
;  6579          6663  2     ! Loop through the Screen Display list.  For each display on the list,
;  6580          6664  2     ! fill the display lines of that display into the appropriate window on
;  6581          6665  2     ! the pasteboard.
;  6582          6666  2     !
;  6583          6667  2     DISPTR = .DBG$SCR_DISPLAY_LIST[0];
;  6584          6668  2     WHILE .DISPTR NEQ DBG$SCR_DISPLAY_LIST DO
;  6585          6669  3         BEGIN
;  6586          6670  3
;  6587          6671  3
;  6588          6672  3         ! Unless the current display is marked as removed from the pasteboard,
;  6589          6673  3         ! we fill it into its window on the pasteboard.
;  6590          6674  3         !
;  6591          6675  3         IF NOT .DISPTR[DBG$V_DISP_REMOVE]
;  6592          6676  3         THEN
;  6593          6677  4             BEGIN
;  6594          6678  4
;  6595          6679  4
;  6596          6680  4             ! Initialize the part of the pasteboard covered by this display to
;  6597          6681  4             ! contain blank lines.  This allows the actual display text to con-
;  6598          6682  4             ! sist of less lines than there are in the corresponding window.
```

```
;  6599          6683  4          !
;  6600          6684  4          SCROLL_AMOUNT = .DISPTR[DBG$W_DISP_SCROLL];
;  6601          6685  4          IF .DISPTR[DBG$V_DISP_INVSCR] THEN SCROLL_AMOUNT = 0;
;  6602          6686  4          RBEG = .DISPTR[DBG$W_DISP_RBEG];
;  6603          6687  4          RLEN = .DISPTR[DBG$W_DISP_RLEN];
;  6604          6688  4          INCR I FROM .RBEG TO MIN(.RBEG + .RLEN, DBG$K_PASTE_SIZE) - 1 DO
;  6605          6689  5              BEGIN
;  6606          6690  5              PASTEBOARD[.I, DBG$B_PASTE_KIND] = DBG$K_PASTE_BLANK;
;  6607          6691  5              PASTEBOARD[.I, DBG$B_PASTE_REND] = .DISPTR[DBG$B_DISP_REND];
;  6608          6692  5              PASTEBOARD[.I, DBG$W_PASTE_SCROLL] = .SCROLL_AMOUNT;
;  6609          6693  5              PASTEBOARD[.I, DBG$L_PASTE_DISPID] = .DISPTR;
;  6610          6694  4              END;
;  6611          6695  4
;  6612          6696  4
;  6613          6697  4          ! Now copy the actual text lines (represented by pointers to
;  6614          6698  4          ! Display Line Entries) from the display into the pasteboard.
;  6615          6699  4          !
;  6616          6700  4          DLEPTR = .DISPTR[DBG$L_DISP_WINDOW_PTR];
;  6617          6701  4          INCR I FROM .RBEG TO MIN(.RBEG + .RLEN, DBG$K_PASTE_SIZE) - 1 DO
;  6618          6702  5              BEGIN
;  6619          6703  5              IF .DLEPTR EQL 0 THEN EXITLOOP;
;  6620          6704  5              IF .DLEPTR EQL DISPTR[DBG$L_DISP_START_LINE_PTR] THEN EXITLOOP;
;  6621          6705  5              PASTEBOARD[.I, DBG$B_PASTE_KIND] = DBG$K_PASTE_TEXT;
;  6622          6706  5              PASTEBOARD[.I, DBG$B_PASTE_REND] = .DLEPTR[DBG$B_DLINE_REND];
;  6623          6707  5              PASTEBOARD[.I, DBG$W_PASTE_SCROLL] = .SCROLL_AMOUNT;
;  6624          6708  5              PASTEBOARD[.I, DBG$L_PASTE_DISPID] = .DISPTR;
;  6625          6709  5              PASTEBOARD[.I, DBG$L_PASTE_DLINE] = .DLEPTR;
;  6626          6710  5              DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  6627          6711  4              END;
;  6628          6712  4
;  6629          6713  4
;  6630          6714  4          ! If this display has Error Line Entries, replace the first part
;  6631          6715  4          ! of the display with the error lines.
;  6632          6716  4          !
;  6633          6717  4          DLEPTR = .DISPTR[DBG$L_DISP_ERROR_PTR];
;  6634          6718  4          INCR I FROM .RBEG TO MIN(.RBEG + .RLEN, DBG$K_PASTE_SIZE) - 1 DO
;  6635          6719  5              BEGIN
;  6636          6720  5              IF .DLEPTR EQL 0 THEN EXITLOOP;
;  6637          6721  5              PASTEBOARD[.I, DBG$B_PASTE_KIND] = DBG$K_PASTE_TEXT;
;  6638          6722  5              PASTEBOARD[.I, DBG$B_PASTE_REND] = .DLEPTR[DBG$B_DLINE_REND];
;  6639          6723  5              PASTEBOARD[.I, DBG$W_PASTE_SCROLL] = .SCROLL_AMOUNT;
;  6640          6724  5              PASTEBOARD[.I, DBG$L_PASTE_DISPID] = .DISPTR;
;  6641          6725  5              PASTEBOARD[.I, DBG$L_PASTE_DLINE] = .DLEPTR;
;  6642          6726  5              DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  6643          6727  4              END;
;  6644          6728  4
;  6645          6729  4
;  6646          6730  4          ! If this display still has line entries on the old-text list,
;  6647          6731  4          ! release them all.  This only occurs for automatically updated
;  6648          6732  4          ! displays when changes are marked and when the display's new
;  6649          6733  4          ! text has less lines than the old text.  Also clear the flag
;  6650          6734  4          ! that marks this display as a new display.
;  6651          6735  4          !
;  6652          6736  4          DISPTR[DBG$V_DISP_NEWDISP] = FALSE;
;  6653          6737  4          DLEPTR = .DISPTR[DBG$L_DISP_OLDTXT_PTR];
;  6654          6738  4          DISPTR[DBG$L_DISP_OLDTXT_PTR] = 0;
;  6655          6739  4          WHILE .DLEPTR NEQ 0 DO
```

```
;  6656          6740  5              BEGIN
;  6657          6741  5              NEXTDLE = .DLEPTR[DBG$L_DLINE_FLINK];
;  6658          6742  5              DBG$REL_MEMORY(.DLEPTR);
;  6659          6743  5              DLEPTR = .NEXTDLE;
;  6660          6744  4              END;
;  6661          6745  4
;  6662          6746  4
;  6663          6747  4          ! Fill in the top border for the display.  We use the display's
;  6664          6748  4          ! label line as the top border.
;  6665          6749  4          !
;  6666          6750  4          PASTEBOARD[.RBEG - 1, DBG$B_PASTE_KIND] = DBG$K_PASTE_LABEL;
;  6667          6751  4          PASTEBOARD[.RBEG - 1, DBG$B_PASTE_REND] = .DISPTR[DBG$B_DISP_REND];
;  6668          6752  4          PASTEBOARD[.RBEG - 1, DBG$L_PASTE_DISPID] = .DISPTR;
;  6669          6753  4
;  6670          6754  4
;  6671          6755  4          ! Fill in the bottom border of the display.  Unless this border
;  6672          6756  4          ! happens to overlay another display, we turn it into the normal
;  6673          6757  4          ! bottom-of-the-screen border.
;  6674          6758  4          !
;  6675          6759  4          IROW = MIN(.RBEG + .RLEN, DBG$K_PASTE_SIZE - 1);
;  6676          6760  4          IF (.PASTEBOARD[.IROW, DBG$B_PASTE_KIND] EQL DBG$K_PASTE_NULL) OR
;  6677          6761  5             (.PASTEBOARD[.IROW, DBG$B_PASTE_KIND] EQL DBG$K_PASTE_BORDER)
;  6678          6762  4          THEN
;  6679          6763  5              BEGIN
;  6680          6764  5              PASTEBOARD[.IROW, DBG$B_PASTE_KIND] = DBG$K_PASTE_BORDER;
;  6681          6765  5              PASTEBOARD[.IROW, DBG$B_PASTE_REND] = .DISPTR[DBG$B_DISP_REND];
;  6682          6766  5              PASTEBOARD[.IROW, DBG$L_PASTE_DISPID] = .DISPTR;
;  6683          6767  5              END
;  6684          6768  5
;  6685          6769  5
;  6686          6770  5              ! But if this border happens to overlay another display, we turn
;  6687          6771  5              ! this line into a label line for the overlayed display.
;  6688          6772  5              !
;  6689          6773  4              ELSE
;  6690          6774  5                  BEGIN
;  6691          6775  5                  PASTEBOARD[.IROW, DBG$B_PASTE_KIND] = DBG$K_PASTE_LABEL;
;  6692          6776  5                  TEMPTR = .PASTEBOARD[.IROW, DBG$L_PASTE_DISPID];
;  6693          6777  5                  PASTEBOARD[.IROW, DBG$B_PASTE_REND] = .TEMPTR[DBG$B_DISP_REND];
;  6694          6778  4                  END;
;  6695          6779  4
;  6696          6780  3              END;                            ! End of code for non-removed displays
;  6697          6781  3
;  6698          6782  3
;  6699          6783  3          ! Clear the scrolling count in the current Display Entry.  Then link
;  6700          6784  3          ! to the next display on the Screen Display List and loop.
;  6701          6785  3          !
;  6702          6786  3          DISPTR[DBG$W_DISP_SCROLL] = 0;
;  6703          6787  3          DISPTR[DBG$V_DISP_INVSCR] = FALSE;
;  6704          6788  3          DISPTR = .DISPTR[DBG$L_DISP_FLINK];
;  6705          6789  3
;  6706          6790  2          END;                                ! End of loop over Screen Display List
;  6707          6791  2
;  6708          6792  2
;  6709          6793  2      ! The pasteboard is updated.  Now return.
;  6710          6794  2      !
;  6711          6795  2      RETURN;
;  6712          6796  2
```

```
;  6713          6797  1     END;


                                        0FFC 00000 DBG$SCR_UPDATE_PASTEBOARD:
                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                ;  6626
                                5E              0C  C2 00002           SUBL2   #12, SP                         ;
                                                50  D4 00005           CLRL    I                               ;  6659
                            51                  50  0C  C5 00007 1$:     MULL3   #12, I, R1                      ;  6660
                                00000000'EF     41  01  90 0000B       MOVB    #1, PASTEBOARD[R1]              ;
                            F0                  50  14  F3 00013       AOBLEQ  #20, I, 1$                      ;
                                            52 00000000'    EF  D0 00017       MOVL    DBG$SCR_DISPLAY_LIST, DISPTR    ;  6667
                                            50 00000000'    EF  9E 0001E 2$:   MOVAB   DBG$SCR_DISPLAY_LIST, R0    ;  6668
                                            50              52  D1 00025       CMPL    DISPTR, R0              ;
                                                            01  12 00028       BNEQ    3$                      ;
                                                                04 0002A       RET                             ;
                                            58      08      A2  9E 0002B 3$:   MOVAB   8(DISPTR), R8           ;  6675
                                            03      02      A8  E9 0002F       BLBC    2(R8), 4$               ;
                                                        0166    31 00033       BRW     20$                     ;
                                            5B      0C      A2  32 00036 4$:   CVTWL   12(DISPTR), SCROLL_AMOUNT    ;  6684
                                02          68              11  E1 0003A       BBC     #17, (R8), 5$           ;  6685
                                                            5B  D4 0003E       CLRL    SCROLL_AMOUNT           ;
                                            53      10      A2  3C 00040 5$:   MOVZWL  16(DISPTR), RBEG        ;  6686
                                        04  AE      12      A2  3C 00044       MOVZWL  18(DISPTR), RLEN        ;  6687
                                        08  AE      04  BE43    9E 00049       MOVAB   @RLEN[RBEG], 8(SP)      ;  6688
                                            57      08      AE  D0 0004F       MOVL    8(SP), R7               ;
                                            15              57  D1 00053       CMPL    R7, #21                 ;
                                                            03  15 00056       BLEQ    6$                      ;
                                            57              15  D0 00058       MOVL    #21, R7                 ;
                                            50      FF      A3  9E 0005B 6$:   MOVAB   -1(R3), I              ;  6692
                                                            22  11 0005F       BRB     8$                      ;
                            51                  50              0C  C5 00061 7$:   MULL3   #12, I, R1          ;  6690
                                            56 00000000'EF      41  9E 00065       MOVAB   PASTEBOARD[R1], R6      ;
                                            66              03  90 0006D       MOVB    #3, (R6)                ;
                                        01  A6      01      A8  90 00070       MOVB    1(R8), 1(R6)            ;  6691
                                        02  A6              5B  B0 00075       MOVW    SCROLL_AMOUNT, 2(R6)    ;  6692
                                            00000000'EF     41  9F 00079       PUSHAB  PASTEBOARD+4[R1]        ;  6693
                                        9E                  52  D0 00080       MOVL    DISPTR, @(SP)+          ;
                            DA                  50              57  F2 00083 8$:   AOBLSS  R7, I, 7$              ;  6688
                                            55      28      A2  D0 00087       MOVL    40(DISPTR), DLEPTR      ;  6700
                                            59      20      A2  9E 0008B       MOVAB   32(R2), R9              ;  6704
                                            50      FF      A3  9E 0008F       MOVAB   -1(R3), I               ;
                                                            38  11 00093       BRB     10$                     ;
                                                            55  D5 00095 9$:   TSTL    DLEPTR                  ;  6703
                                                            38  13 00097       BEQL    11$                     ;
                                            59              55  D1 00099       CMPL    DLEPTR, R9              ;  6704
                                                            33  13 0009C       BEQL    11$                     ;
                            51                  50              0C  C5 0009E       MULL3   #12, I, R1          ;  6705
                                            56 00000000'EF      41  9E 000A2       MOVAB   PASTEBOARD[R1], R6      ;
                                            66              02  90 000AA       MOVB    #2, (R6)                ;
                                        01  A6      08      A5  90 000AD       MOVB    8(DLEPTR), 1(R6)        ;  6706
                                        02  A6              5B  B0 000B2       MOVW    SCROLL_AMOUNT, 2(R6)    ;  6707
                                            00000000'EF     41  9F 000B6       PUSHAB  PASTEBOARD+4[R1]        ;  6708
                                        9E                  52  D0 000BD       MOVL    DISPTR, @(SP)+          ;
                                            00000000'EF     41  9F 000C0       PUSHAB  PASTEBOARD+8[R1]        ;  6709
```

```
                9E              55   D0 000C7          MOVL    DLEPTR, @(SP)+                          ;
                55              65   D0 000CA          MOVL    (DLEPTR), DLEPTR                        ; 6710
        C4      50              57   F2 000CD 10$:     AOBLSS  R7, I, 9$                               ; 6701
                55        2C    A2   D0 000D1 11$:     MOVL    44(DISPTR), DLEPTR                      ; 6717
                50        FF    A3   9E 000D5          MOVAB   -1(R3), I                               ; 6718
                                33   11 000D9          BRB     13$
                                55   D5 000DB 12$:     TSTL    DLEPTR                                  ; 6720
                                33   13 000DD          BEQL    14$
        51      50              0C   C5 000DF          MULL3   #12, I, R1                              ; 6721
                59 00000000'EF  41   9E 000E3          MOVAB   PASTEBOARD[R1], R9
                69              02   90 000EB          MOVB    #2, (R9)
                01    A9        08   A5 000EE          MOVB    8(DLEPTR), 1(R9)                        ; 6722
                02    A9              5B   B0 000F3    MOVW    SCROLL_AMOUNT, 2(R9)                    ; 6723
                   00000000'EF  41   9F 000F7          PUSHAB  PASTEBOARD+4[R1]                        ; 6724
                9E              52   D0 000FE          MOVL    DISPTR, @(SP)+
                   00000000'EF  41   9F 00101          PUSHAB  PASTEBOARD+8[R1]                        ; 6725
                9E              55   D0 00108          MOVL    DLEPTR, @(SP)+
                55              65   D0 0010B          MOVL    (DLEPTR), DLEPTR                        ; 6726
        C9      50              57   F2 0010E 13$:     AOBLSS  R7, I, 12$                              ; 6718
                02    A8        20   8A 00112 14$:     BICB2   #32, 2(R8)                              ; 6736
                55        30    A2   D0 00116          MOVL    48(DISPTR), DLEPTR                      ; 6737
                          30    A2   D4 0011A          CLRL    48(DISPTR)                              ; 6738
                                55   D5 0011D 15$:     TSTL    DLEPTR                                  ; 6739
                                11   13 0011F          BEQL    16$
                6E              65   D0 00121          MOVL    (DLEPTR), NEXTDLE                       ; 6741
                                55   DD 00124          PUSHL   DLEPTR                                  ; 6742
    00000000G   00              01   FB 00126          CALLS   #1, DBG$REL_MEMORY
                55              6E   D0 0012D          MOVL    NEXTDLE, DLEPTR                         ; 6743
                                EB   11 00130          BRB     15$                                     ; 6739
        50      53              0C   C5 00132 16$:     MULL3   #12, RBEG, R0                           ; 6750
   00000000'EF  40              04   90 00136          MOVB    #4, PASTEBOARD-12[R0]
   00000000'EF  40        01    A8   90 0013E          MOVB    1(R8), PASTEBOARD-11[R0]                ; 6751
                   00000000'EF  40   9F 00147          PUSHAB  PASTEBOARD-8[R0]                        ; 6752
                9E              52   D0 0014E          MOVL    DISPTR, @(SP)+
                50        08    AE   D0 00151          MOVL    8(SP), R0                               ; 6759
                14              50   D1 00155          CMPL    R0, #20
                                03   15 00158          BLEQ    17$
                50              14   D0 0015A          MOVL    #20, R0
                54              50   D0 0015D 17$:     MOVL    R0, IROW
        50      54              0C   C5 00160          MULL3   #12, IROW, R0                           ; 6760
                51 00000000'EF  40   9E 00164          MOVAB   PASTEBOARD[R0], R1
                01              61   91 0016C          CMPB    (R1), #1
                                05   13 0016F          BEQL    18$
                05              61   91 00171          CMPB    (R1), #5                                ; 6761
                                14   12 00174          BNEQ    19$
                61              05   90 00176 18$:     MOVB    #5, (R1)                                ; 6764
          01    A1        01    A8   90 00179          MOVB    1(R8), 1(R1)                            ; 6765
                   0u000000'EF  40   9F 0017E          PUSHAB  PASTEBOARD+4[R0]                        ; 6766
                9E              52   D0 00185          MOVL    DISPTR, @(SP)+
                                12   11 00188          BRB     20$                                     ; 6760
                61              04   90 0018A 19$:     MOVB    #4, (R1)                                ; 6775
                   00000000'EF  40   9F 0018D          PUSHAB  PASTEBOARD+4[R0]                        ; 6776
                5A              9E   D0 00194          MOVL    @(SP)+, TEMPTR
          01    A1        09    AA   90 00197          MOVB    9(TEMPTR), 1(R1)                        ; 6777
                          0C    A2   B4 0019C 20$:     CLRW    12(DISPTR)                              ; 6786
          02    A8              02   8A 0019F          BICB2   #2, 2(R8)                               ; 6787
                52              62   D0 001A3          MOVL    (DISPTR), DISPTR                        ; 6788
```

```
                    FE75  31 001A6         BRW     2$                      ; 6668
                          04 001A9         RET                             ; 6797

; Routine Size:  426 bytes,    Routine Base:  DBG$CODE + 38A6
```

```
;6715     6798  1 GLOBAL ROUTINE DBG$SCR_WRITE_ERROR(RABPTR) =
;6716     6799  1 !
;6717     6800  1 ! FUNCTION
;6718     6801  1 !       This routine writes an error message line to a screen display.  It is
;6719     6802  1 !       called from routine DBG$OUT_MESSAGE when an error has been signalled
;6720     6803  1 !       and the corresponding message is about to be output.  If the global
;6721     6804  1 !       variable DBG$GL_SCREEN_ERROR is non-zero, this routine is called in-
;6722     6805  1 !       stead of the $PUT system service to output the message.  It does so
;6723     6806  1 !       by attaching the error message text to the Screen Display Entry that
;6724     6807  1 !       DBG$GL_SCREEN_ERROR points to.
;6725     6808  1 !
;6726     6809  1 !       Error messages are only directed to screen displays when the contents
;6727     6810  1 !       of automatically generated displays are generated by invoking the DEBUG
;6728     6811  1 !       command lists associated with such displays.  This allows the messages
;6729     6812  1 !       associated with such displays to be displayed in the display's them-
;6730     6813  1 !       selves, rather than being output to the input window at the bottom of
;6731     6814  1 !       the screen.  Such messages would be quite confusing at the bottom of
;6732     6815  1 !       the screen since they are not associated with the current DEBUG command
;6733     6816  1 !       entered from the terminal.
;6734     6817  1 !
;6735     6818  1 ! INPUTS
;6736     6819  1 !       RABPTR  - A pointer to an RMS Record Access Block (RAB) which describes
;6737     6820  1 !                 the line of error message text to be written to the currently
;6738     6821  1 !                 active error display.
;6739     6822  1 !
;6740     6823  1 ! OUTPUTS
;6741     6824  1 !       This routine returns TRUE as its value if the error message was success-
;6742     6825  1 !                 fully written to the current output display.  If the write
;6743     6826  1 !                 did not succeed, the routine returns FALSE to indicate that
;6744     6827  1 !                 the error message should be forced out to the user's terminal
;6745     6828  1 !                 using the $PUT system service.
;6746     6829  1 !
;6747     6830  1
;6748     6831  2     BEGIN
;6749     6832  2
;6750     6833  2     MAP
;6751     6834  2         RABPTR: REF BLOCK[,BYTE];          ! Pointer to Record Access Block
;6752     6835  2
;6753     6836  2     LOCAL
;6754     6837  2         DISPTR: REF DBG$DISP_ENTRY,        ! Pointer to Screen Display Entry
;6755     6838  2         DLEPTR: REF DBG$DLINE_ENTRY,       ! Pointer to the Display Line Entry for
;6756     6839  2                                            !     the new output line
;6757     6840  2         DLINE_TEXT: REF VECTOR[,BYTE],     ! Pointer to text buffer in Line Entry
;6758     6841  2         MSGEND,                            ! Pointer to first byte after message
;6759     6842  2         MSGLEN,                            ! The length of the message text
;6760     6843  2         MSGPTR: REF VECTOR[,BYTE],         ! The address of the message text
;6761     6844  2         NXTMSGPTR,                         ! Pointer to rest of message text--this
;6762     6845  2                                            !     is advanced as message is broken
;6763     6846  2                                            !     into separate lines
;6764     6847  2         PREVPTR: REF DBG$DLINE_ENTRY,      ! Pointer to previous Display Line
;6765     6848  2                                            !     Entry on error message list
;6766     6849  2         TEMPBUF: VECTOR[256, BYTE],        ! Temporary buffer for de-tabbed text
;6767     6850  2         TLEN;                              ! The length of the de-tabbed text
;6768     6851  2
;6769     6852  2     ENABLE
;6770     6853  2         DBG$FINAL_HANDL;                   ! Enable the Final Handler during the
;6771     6854  2                                            !     processing of this error
```

```
; 6772          6855  2
; 6773          6856  2
; 6774          6857  2
; 6775          6858  2      ! Pick up the pointer to the selected Screen Display Entry to receive the
; 6776          6859  2      ! error message.  Then clear DBG$GL_SCREEN_ERROR so that any error in the
; 6777          6860  2      ! processing of this error does not cause this routine to be re-entered.
; 6778          6861  2      ! (Any such secondary error will thus be output using $PUT.)
; 6779          6862  2      !
; 6780          6863  2      DISPTR = .DBG$GL_SCREEN_ERROR;
; 6781          6864  2      DBG$GL_SCREEN_ERROR = 0;
; 6782          6865  2
; 6783          6866  2
; 6784          6867  2      ! Extract the message text's address and length from the RAB.
; 6785          6868  2      !
; 6786          6869  2      NXTMSGPTR = .RABPTR[RAB$L_RBF];
; 6787          6870  2      MSGLEN = .RABPTR[RAB$W_RSZ];
; 6788          6871  2      MSGEND = .NXTMSGPTR + .MSGLEN;
; 6789          6872  2
; 6790          6873  2
; 6791          6874  2      ! Set up a loop through the lines of the error message.  This loop loops
; 6792          6875  2      ! through the individual lines, separated by carriage-return/line-feeds,
; 6793          6876  2      ! within the message text.  Each such line must be output separately to
; 6794          6877  2      ! the appropriate screen display in order for the lines to be positioned
; 6795          6878  2      ! correctly on the screen.
; 6796          6879  2      !
; 6797          6880  2      WHILE .NXTMSGPTR LSS .MSGEND DO
; 6798          6881  3          BEGIN
; 6799          6882  3
; 6800          6883  3
; 6801          6884  3          ! Scan the message text for any carriage returns and line feeds.  When
; 6802          6885  3          ! a carriage return or carriage-return/line-feed is found, we break the
; 6803          6886  3          ! message text into two lines so that we can output the first line and
; 6804          6887  3          ! remember where the rest of the message starts.  If no carriage-return
; 6805          6888  3          ! is found, we output the entire remaining message as one line.
; 6806          6889  3          !
; 6807          6890  3          MSGPTR = .NXTMSGPTR;
; 6808          6891  3          MSGLEN = .MSGEND - .MSGPTR;
; 6809          6892  3          NXTMSGPTR = .MSGEND;
; 6810          6893  3          INCR I FROM 0 TO .MSGLEN - 1 DO
; 6811          6894  4              BEGIN
; 6812          6895  4              IF .MSGPTR[.I] EQL DBG$K_CAR_RETURN
; 6813          6896  4              THEN
; 6814          6897  5                  BEGIN
; 6815          6898  5                  MSGLEN = .I;
; 6816          6899  5                  NXTMSGPTR = MSGPTR[.I + 1];
; 6817          6900  5                  IF .MSGPTR[.I + 1] EQL DBG$K_LINE_FEED
; 6818          6901  5                  THEN
; 6819          6902  5                      NXTMSGPTR = MSGPTR[.I + 2];
; 6820          6903  5
; 6821          6904  5                  EXITLOOP;
; 6822          6905  4                  END;
; 6823          6906  4
; 6824          6907  3              END;
; 6825          6908  3
; 6826          6909  3
; 6827          6910  3          ! We now have one line of message text, given by MSGPTR and MSGLEN.
; 6828          6911  3          ! This is the message to be output this time through the WHILE loop.
```

```
; 6829          6912  3          !
; 6830          6913  3          ! If the error message is being directed to a normal display which
; 6831          6914  3          ! is automatically generated with a DO command list, simply cal
; 6832          6915  3          ! WRITE_LINE to output the error message to the desired display.
; 6833          6916  3          !
; 6834          6917  3          IF .DISPTR[DBG$B_DISP_KIND] EQL DBG$K_DISP_DO
; 6835          6918  3          THEN
; 6836          6919  3              DBG$SCR_WRITE_LINE(.MSGLEN, .MSGPTR, .DISPTR)
; 6837          6920  3
; 6838          6921  3
; 6839          6922  3          ! If we do not have a SOURCE display at this point, we simply return
; 6840          6923  3          ! FALSE to cause the error message to be output the normal way (with
; 6841          6924  3          ! $PUT).
; 6842          6925  3          !
; 6843          6926  3          ELSE IF (.DISPTR[DBG$B_DISP_KIND] NEQ DBG$K_DISP_SOURCE) OR
; 6844          6927  4                  (.DISPTR[DBG$L_DISP_CMDLIST] EQL 0)
; 6845          6928  3          THEN
; 6846          6929  4              BEGIN
; 6847          6930  4              DBG$GL_SCREEN_ERROR = .DISPTR;
; 6848          6931  4              RETURN FALSE;
; 6849          6932  4              END
; 6850          6933  4
; 6851          6934  4
; 6852          6935  4          ! The error message is associated with a source display.  Massage the
; 6853          6936  4          ! text by expanding all tabs to blanks.  This is necessary to make
; 6854          6937  4          ! subsequent length computations and scrolling to the right work
; 6855          6938  4          ! correctly.  The expanded text is copied into the TEMPBUF temporary
; 6856          6939  4          ! buffer.
; 6857          6940  4          !
; 6858          6941  3          ELSE
; 6859          6942  4              BEGIN
; 6860          6943  4              TLEN = 0;
; 6861          6944  4              INCR I FROM 0 TO .MSGLEN - 1 DO
; 6862          6945  5                  BEGIN
; 6863          6946  5
; 6864          6947  5
; 6865          6948  5                  ! If the current character is a tab character, expand it to
; 6866          6949  5                  ! blanks.
; 6867          6950  5                  !
; 6868          6951  5                  IF .MSGPTR[.I] EQL TAB_CHAR
; 6869          6952  5                  THEN
; 6870          6953  6                      BEGIN
; 6871          6954  6                      INCR J FROM 1 TO 8 - (.TLEN MOD 8) DO
; 6872          6955  7                          BEGIN
; 6873          6956  7                          TEMPBUF[.TLEN] = ' ';
; 6874          6957  7                          TLEN = .TLEN + 1;
; 6875          6958  7                          IF .TLEN GEQ 255 THEN EXITLOOP;
; 6876          6959  6                          END;
; 6877          6960  6
; 6878          6961  6                      END
; 6879          6962  6
; 6880          6963  6
; 6881          6964  6                  ! If the current character is not a tab, just copy it as is.
; 6882          6965  6                  !
; 6883          6966  5                  ELSE
; 6884          6967  6                      BEGIN
; 6885          6968  6                      TEMPBUF[.TLEN] = .MSGPTR[.I];
```

```
;  6886          6969  6                  TLEN = .TLEN + 1;
;  6887          6970  5                  END;
;  6888          6971  5
;  6889          6972  5              IF .TLEN GEQ 255 THEN EXITLOOP;
;  6890          6973  4              END;
;  6891          6974  4
;  6892          6975  4
;  6893          6976  4          ! Allocate a Screen Display Line Entry for the new error message.
;  6894          6977  4          ! Then fill in the fields of the Display Line Entry, including the
;  6895          6978  4          ! new text, and link that entry into the source display's Error
;  6896          6979  4          ! Line Entry List.  Note that we highlight the error message with
;  6897          6980  4          ! reverse video and bolding.
;  6898          6981  4          !
;  6899          6982  4          DLEPTR = DBG$GET_MEMORY(DBG$K_DLINE_ENTSIZE + .TLEN/%UPVAL + 1);
;  6900          6983  4          DLEPTR[DBG$B_DLINE_REND] = .DISPTR[DBG$B_DISP_REND] XOR
;  6901          6984  4                                  (DBG$M_DISP_REND_RV OR DBG$M_DISP_REND_BLD);
;  6902          6985  4          DLEPTR[DBG$B_DLINE_LENGTH] = .TLEN;
;  6903          6986  4          DLINE_TEXT = DLEPTR[DBG$A_DLINE_TEXT];
;  6904          6987  4          DLINE_TEXT[0] = .TLEN;
;  6905          6988  4          CH$MOVE(.TLEN, TEMPBUF, DLINE_TEXT[1]);
;  6906          6989  4          PREVPTR = DISPTR[DBG$L_DISP_ERROR_PTR];
;  6907          6990  4          WHILE .PREVPTR[DBG$L_DLINE_FLINK] NEQ 0 DO
;  6908          6991  4              PREVPTR = .PREVPTR[DBG$L_DLINE_FLINK];
;  6909          6992  4
;  6910          6993  4          PREVPTR[DBG$L_DLINE_FLINK] = .DLEPTR;
;  6911          6994  4
;  6912          6995  3          END;                            ! End of source display ELSE clause
;  6913          6996  3
;  6914          6997  2      END;                                ! End of WHILE loop over error lines
;  6915          6998  2
;  6916          6999  2
;  6917          7000  2      ! The error message has been successfully written to the error display.
;  6918          7001  2      ! Now restore DBG$GL_SCREEN_ERROR and return.
;  6919          7002  2      !
;  6920          7003  2      DBG$GL_SCREEN_ERROR = .DISPTR;
;  6921          7004  2      RETURN TRUE;
;  6922          7005  2
;  6923          7006  1      END;


                                    0FFC 00000          .ENTRY  DBG$SCR_WRITE_ERROR, Save R2,R3,R4,R5,R6,-  ; 6798
                                                                R7,R8,R9,R10,R11
                                                                                                           ;
            5E      FEF4    CE  9E 00002                MOVAB   -268(SP), SP                               ;
            6D      0121    CF  DE 00007                MOVAL   20$, (FP)                                  ; 6831
            57 00000000'    EF  D0 0000C                MOVL    DBG$GL_SCREEN_ERROR, DISPTR                ; 6863
               00000000'    EF  D4 00013                CLRL    DBG$GL_SCREEN_ERROR                        ; 6864
            50        04    AC  D0 00019                MOVL    RABPTR, R0                                 ; 6869
        08  AE        28    A0  D0 0001D                MOVL    40(R0), NXTMSGPTR                          ;
            59        22    A0  3C 00022                MOVZWL  34(R0), MSGLEN                             ; 6870
        04  AE        08 BE49  9E 00026                MOVAB   @NXTMSGPTR[MSGLEN], MSGEND                 ; 6871
        04  AE        08    AE  D1 0002C 1$:            CMPL    NXTMSGPTR, MSGEND                          ; 6880
                          03    19 00031                BLSS    2$                                         ;
                      00E8    31 00033                BRW     18$                                        ;
            5A        08    AE  D0 00036 2$:            MOVL    NXTMSGPTR, MSGPTR                          ; 6890
```

```
                59       04   AE              5A   C3 0003A              SUBL3   MSGPTR, MSGEND, MSGLEN                  ; 6891
                         08   AE        04    AE   D0 0003F              MOVL    MSGEND, NXTMSGPTR                       ; 6892
                              51          59   D0 00044              MOVL    MSGLEN, R1                              ; 6893
                              50          01   CE 00047              MNEGL   #1, I                                   ;
                                          1E   11 0004A              BRB     4$                                      ;
                              0D        604A   91 0004C 3$:          CMPB    (I)[MSGPTR], #13                        ; 6895
                                          18   12 00050              BNEQ    4$                                      ;
                              59          50   D0 00052              MOVL    I, MSGLEN                               ; 6898
                         08   AE    01 A04A   9E 00055              MOVAB   1(I)[MSGPTR], NXTMSGPTR                 ; 6899
                              0A    01 A04A   91 0005B              CMPB    1(I)[MSGPTR], #10                       ; 6900
                                          0C   12 00060              BNEQ    5$                                      ;
                         08   AE    02 A04A   9E 00062              MOVAB   2(I)[MSGPTR], NXTMSGPTR                 ; 6902
                                          04   11 00068              BRB     5$                                      ; 6897
                DE            50          51   F2 0006A 4$:          AOBLSS  R1, I, 3$                               ; 6893
                              02        08   A7   91 0006E 5$:          CMPB    8(DISPTR), #2                           ; 6917
                                          0C   12 00072              BNEQ    6$                                      ;
                                          57   DD 00074              PUSHL   DISPTR                                  ; 6919
                              7E          59   7D 00076              MOVQ    MSGLEN, -(SP)                           ;
                       0000V  CF          03   FB 00079              CALLS   #3, DBG$SCR_WRITE_LINE                  ;
                                          AC   11 0007E              BRB     1$                                      ;
                              03        08   A7   91 00080 6$:          CMPB    8(DISPTR), #3                           ; 6926
                                          05   12 00084              BNEQ    7$                                      ;
                                        48   A7   D5 00086              TSTL    72(DISPTR)                              ; 6927
                                          0A   12 00089              BNEQ    8$                                      ;
                  00000000'   EF          57   D0 0008B 7$:          MOVL    DISPTR, DBG$GL_SCREEN_ERROR             ; 6930
                                        0094   31 00092              BRW     19$                                     ; 6931
                                          56   D4 00095 8$:          CLRL    TLEN                                    ; 6943
                              52          01   CE 00097              MNEGL   #1, I                                   ; 6944
                                          3F   11 0009A              BRB     14$                                     ;
                              09        624A   91 0009C 9$:          CMPB    (I)[MSGPTR], #9                         ; 6951
                                          28   12 000A0              BNEQ    12$                                     ;
                7E            00          56   01   7A 000A2              EMUL    #1, TLEN, #0, -(SP)                     ; 6954
                50            50          8E   08   7B 000A7              EDIV    #8, (SP)+, R0, R0                       ;
                              50          08   50   C3 000AC              SUBL3   R0, #8, R0                              ;
                                          51   D4 000B0              CLRL    J                                       ;
                                          10   11 000B2              BRB     11$                                     ;
                                    0C AE46        20   90 000B4 10$:          MOVB    #32, TEMPBUF[TLEN]                      ; 6956
                                          56   D6 000B9              INCL    TLEN                                    ; 6957
                         000000FF     8F        56   D1 000BB              CMPL    TLEN, #255                              ; 6958
                                          0E   18 000C2              BGEQ    13$                                     ;
                              EC          51   50   F3 000C4 11$:          AOBLEQ  R0, J, 10$                      ; 6954
                                          08   11 000C8              BRB     13$                                     ; 6951
                                    0C AE46      624A   90 000CA 12$:          MOVB    (I)[MSGPTR], TEMPBUF[TLEN]              ; 6968
                                          56   D6 000D0              INCL    TLEN                                    ; 6969
                         000000FF     8F        56   D1 000D2 13$:          CMPL    TLEN, #255                              ; 6972
                                          04   18 000D9              BGEQ    15$                                     ;
                              BD          52          59   F2 000DB 14$:          AOBLSS  MSGLEN, I, 9$                   ; 6944
                              50          56          04   C7 000DF 15$:          DIVL3   #4, TLEN, R0                    ; 6982
                                                04    A0   9F 000E3              PUSHAB  4(R0)                           ;
                         00000000G    00          01   FB 000E6              CALLS   #1, DBG$GET_MEMORY              ;
                                      5B          50   D0 000ED              MOVL    R0, DLEPTR                      ;
                08    AB        09    A7          03   8D 000F0              XORB3   #3, 9(DISPTR), 8(DLEPTR)        ; 6984
                                    0A    AB        56   90 000F6              MOVB    TLEN, 10(DLEPTR)                ; 6985
                                          6E  0C    AB   9E 000FA              MOVAB   12(R11), DLINE_TEXT             ; 6986
                                    00    BE        56   90 000FE              MOVB    TLEN, @DLINE_TEXT               ; 6987
                              7E          6E          01   C1 00102              ADDL3   #1, DLINE_TEXT, -(SP)           ; 6988
                              9E        10    AE        56   28 00106              MOVC3   TLEN, TEMPBUF, @(SP)+           ;
```

```
                         58        2C    A7  9E 0010B                MOVAB   44(R7), PREVPTR                 ; 6989
                                         68  D5 0010F 16$:           TSTL    (PREVPTR)                       ; 6990
                                         05  13 00111                BEQL    17$                             ;
                         58              68  D0 00113                MOVL    (PREVPTR), PREVPTR              ; 6991
                                         F7  11 00116                BRB     16$                             ;
                         68              5B  D0 00118 17$:           MOVL    DLEPTR, (PREVPTR)               ; 6993
                                      FF0E   31 0011B                BRW     1$                              ; 6880
                00000000'  EF            57  D0 0011E 18$:           MOVL    DISPTR, DBG$GL_SCREEN_ERROR     ; 7003
                         50              01  D0 00125                MOVL    #1, R0                          ; 7004
                                             04 00128                RET                                     ;
                                         50  D4 00129 19$:           CLRL    R0                              ; 7006
                                             04 0012B                RET                                     ;
                                       0000  0012C 20$:              .WORD   Save nothing                    ; 6831
                                         7E  D4 0012E                CLRL    -(SP)                           ;
                                         5E  DD 00130                PUSHL   SP                              ;
                         7E        04    AC  7D 00132                MOVQ    4(AP), -(SP)                    ;
                00000000G  00            03  FB 00136                CALLS   #3, DBG$FINAL_HANDL             ;
                                             04 0013D                RET                                     ;

; Routine Size:  318 bytes,    Routine Base:  DBG$CODE + 3A50
```

```
;  6925          7007  1 GLOBAL ROUTINE DBG$SCR_WRITE_LINE(LENGTH, TEXTPTR, DISPTR): NOVALUE =
;  6926          7008  1 !
;  6927          7009  1 ! FUNCTION
;  6928          7010  1 !       This routine writes a line of DEBUG output to the current output
;  6929          7011  1 !       display or to a specified screen display.  A new Display Line
;  6930          7012  1 !       Entry is allocated for the new text line and is then appended to
;  6931          7013  1 !       the list of display lines assocated with this display.
;  6932          7014  1 !
;  6933          7015  1 ! INPUTS
;  6934          7016  1 !       LENGTH  - The length of the text line, i.e. the number of bytes to be
;  6935          7017  1 !                 retrieved from the TEXTPTR location.
;  6936          7018  1 !
;  6937          7019  1 !       TEXTPTR - A pointer to the first byte of the ASCII text which con-
;  6938          7020  1 !                 stitutes the line to be written to the output display.
;  6939          7021  1 !
;  6940          7022  1 !       DISPTR  - The Display ID (Display Entry pointer) of the display to
;  6941          7023  1 !                 which this line of output is to be written.
;  6942          7024  1 !
;  6943          7025  1 ! OUTPUTS
;  6944          7026  1 !       NONE
;  6945          7027  1 !
;  6946          7028  1
;  6947          7029  2     BEGIN
;  6948          7030  2
;  6949          7031  2     MAP
;  6950          7032  2         TEXTPTR: REF VECTOR[,BYTE],     ! Pointer to ASCII text of output line
;  6951          7033  2         DISPTR: REF DBG$DISP_ENTRY;     ! Pointer to output Screen Display Entry
;  6952          7034  2
;  6953          7035  2     LOCAL
;  6954          7036  2         BLINK: REF DBG$DLINE_ENTRY,     ! Pointer to previous Display Line Entry
;  6955          7037  2         CHANGE_FLAG,                    ! Flag set if line contents changed
;  6956          7038  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to the Display Line Entry for
;  6957          7039  2                                         !     the new output line
;  6958          7040  2         DLINE_TEXT: REF VECTOR[,BYTE],  ! Pointer to text buffer in Line Entry
;  6959          7041  2         FLINK: REF DBG$DLINE_ENTRY,     ! Pointer to next Display Line Entry
;  6960          7042  2         NEW_DROW,                       ! New DROW value if scrolling to bottom
;  6961          7043  2         OLD_DROW,                       ! Old DROW value if scrolling to bottom
;  6962          7044  2         OLDPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Display Line Entry for old
;  6963          7045  2                                         !     text--used to mark changes
;  6964          7046  2         OLDTXT: REF VECTOR[,BYTE],      ! Pointer to line's old text buffer
;  6965          7047  2         TEMPBUF: VECTOR[256, BYTE],     ! Temporary buffer for de-tabbed text
;  6966          7048  2         TEMP_PTR: REF DBG$DLINE_ENTRY,  ! Temporary pointer to Display Line Entry
;  6967          7049  2         TLEN,                           ! The length of the de-tabbed text
;  6968          7050  2         WSPTR: REF DBG$DLINE_ENTRY;     ! Pointer to window start line (points
;  6969          7051  2                                         !     to a Display Line Entry)
;  6970          7052  2
;  6971          7053  2
;  6972          7054  2
;  6973          7055  2     ! Check that the input parameters are valid.
;  6974          7056  2     !
;  6975          7057  2     IF .LENGTH LSS 0 THEN $DBG_ERROR('DBGSCREEN\WRITE_LINE 10');
;  6976          7058  2     IF .DISPTR EQL 0 THEN $DBG_ERROR('DBGSCREEN\WRITE_LINE 20');
;  6977          7059  2
;  6978          7060  2
;  6979          7061  2     ! Expand all tabs to blanks.  This is necessary in order to make subsequent
;  6980          7062  2     ! length computations and scrolling to the right work correctly.  The
;  6981          7063  2     ! expanded text is copied into the TEMPBUF temporary buffer.
```

```
;  6982          7064  2     !
;  6983          7065  2     TLEN = 0;
;  6984          7066  2     INCR I FROM 0 TO .LENGTH - 1 DO
;  6985          7067  3         BEGIN
;  6986          7068  3
;  6987          7069  3
;  6988          7070  3         ! If the current character is a tab character, expand it to blanks.
;  6989          7071  3         !
;  6990          7072  3         IF .TEXTPTR[.I] EQL TAB_CHAR
;  6991          7073  3         THEN
;  6992          7074  4             BEGIN
;  6993          7075  4             INCR J FROM 1 TO 8 - (.TLEN MOD 8) DO
;  6994          7076  5                 BEGIN
;  6995          7077  5                 TEMPBUF[.TLEN] = ' ';
;  6996          7078  5                 TLEN = .TLEN + 1;
;  6997          7079  5                 IF .TLEN GEQ 255 THEN EXITLOOP;
;  6998          7080  4                 END;
;  6999          7081  4
;  7000          7082  4             END
;  7001          7083  4
;  7002          7084  4
;  7003          7085  4         ! If the current character is not a tab, just copy it as is.
;  7004          7086  4         !
;  7005          7087  3         ELSE
;  7006          7088  4             BEGIN
;  7007          7089  4             TEMPBUF[.TLEN] = .TEXTPTR[.I];
;  7008          7090  4             TLEN = .TLEN + 1;
;  7009          7091  3             END;
;  7010          7092  3
;  7011          7093  3         IF .TLEN GEQ 255 THEN EXITLOOP;
;  7012          7094  2         END;
;  7013          7095  2
;  7014          7096  2
;  7015          7097  2     ! If the display is already full, meaning that it has the maximum number
;  7016          7098  2     ! of text lines it is allowed to have, we remove the first line (the oldest
;  7017          7099  2     ! line) of the display.  Note, however, that we reuse that same Display
;  7018          7100  2     ! Line Entry if it is large enough to hold the new text line.
;  7019          7101  2     !
;  7020          7102  2     DLEPTR = 0;
;  7021          7103  2     IF .DISPTR[DBG$W_DISP_LINECNT] EQL .DISPTR[DBG$W_DISP_MAX_LINECNT]
;  7022          7104  2     THEN
;  7023          7105  3         BEGIN
;  7024          7106  3
;  7025          7107  3
;  7026          7108  3         ! Unlink this entry from the doubly linked Display Line Entry list
;  7027          7109  3         ! associated with the DISPTR Screen Display Entry.
;  7028          7110  3         !
;  7029          7111  3         DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  7030          7112  3         FLINK = .DLEPTR[DBG$L_DLINE_FLINK];
;  7031          7113  3         BLINK = .DLEPTR[DBG$L_DLINE_BLINK];
;  7032          7114  3         FLINK[DBG$L_DLINE_BLINK] = .BLINK;
;  7033          7115  3         BLINK[DBG$L_DLINE_FLINK] = .FLINK;
;  7034          7116  3         DISPTR[DBG$W_DISP_LINECNT] = .DISPTR[DBG$W_DISP_LINECNT] - 1;
;  7035          7117  3
;  7036          7118  3
;  7037          7119  3         ! Adjust the window start pointer and DROW if necessary to account
;  7038          7120  3         ! for the fact that the first line of the display is being removed.
```

```
;  7039     7121  3          !
;  7040     7122  3          IF .DISPTR[DBG$W_DISP_DROW] GTR 1
;  7041     7123  3          THEN
;  7042     7124  3              DISPTR[DBG$W_DISP_DROW] = .DISPTR[DBG$W_DISP_DROW] - 1
;  7043     7125  3
;  7044     7126  3          ELSE
;  7045     7127  4              BEGIN
;  7046     7128  4              IF .DISPTR[DBG$W_DISP_LINECNT] EQL 0 THEN FLINK = 0;
;  7047     7129  4              DISPTR[DBG$L_DISP_WINDOW_PTR] = .FLINK;
;  7048     7130  4              DISPTR[DBG$W_DISP_SCROLL] = .DISPTR[DBG$W_DISP_SCROLL] + 1;
;  7049     7131  3              END;
;  7050     7132  3
;  7051     7133  3
;  7052     7134  3          ! If the discarded Display Line Entry is too small to hold the new
;  7053     7135  3          ! text line, release it to the memory pool and clear DLEPTR to zero.
;  7054     7136  3          ! However, if it is big enough, we keep it and DLEPTR points to it.
;  7055     7137  3          !
;  7056     7138  3          IF .TLEN GTR .DLEPTR[DBG$B_DLINE_LENGTH]
;  7057     7139  3          THEN
;  7058     7140  4              BEGIN
;  7059     7141  4              DBG$REL_MEMORY(.DLEPTR);
;  7060     7142  4              DLEPTR = 0;
;  7061     7143  3              END;
;  7062     7144  3
;  7063     7145  2          END;
;  7064     7146  2
;  7065     7147  2
;  7066     7148  2      ! Allocate a Screen Display Line Entry for the new text line unless we
;  7067     7149  2      ! found that we can reuse the line entry for the discarded oldest line.
;  7068     7150  2      !
;  7069     7151  2      IF .DLEPTR EQL 0
;  7070     7152  2      THEN
;  7071     7153  3          BEGIN
;  7072     7154  3          DLEPTR = DBG$GET_MEMORY(DBG$K_DLINE_ENTSIZE + .TLEN/%UPVAL + 1);
;  7073     7155  3          DLEPTR[DBG$B_DLINE_LENGTH] = .TLEN;
;  7074     7156  2          END;
;  7075     7157  2
;  7076     7158  2
;  7077     7159  2      ! Fill in the fields of the Display Line Entry, including the new text, and
;  7078     7160  2      ! then link that entry into the display's line entry list.
;  7079     7161  2      !
;  7080     7162  2      DLEPTR[DBG$B_DLINE_REND] = .DISPTR[DBG$B_DISP_REND];
;  7081     7163  2      DLINE_TEXT = DLEPTR[DBG$A_DLINE_TEXT];
;  7082     7164  2      DLINE_TEXT[0] = .TLEN;
;  7083     7165  2      CH$MOVE(.TLEN, TEMPBUF, DLINE_TEXT[1]);
;  7084     7166  2      FLINK = DISPTR[DBG$L_DISP_START_LINE_PTR];
;  7085     7167  2      BLINK = .DISPTR[DBG$L_DISP_END_LINE_PTR];
;  7086     7168  2      DLEPTR[DBG$L_DLINE_FLINK] = .FLINK;
;  7087     7169  2      DLEPTR[DBG$L_DLINE_BLINK] = .BLINK;
;  7088     7170  2      FLINK[DBG$L_DLINE_BLINK] = .DLEPTR;
;  7089     7171  2      BLINK[DBG$L_DLINE_FLINK] = .DLEPTR;
;  7090     7172  2      DISPTR[DBG$W_DISP_LINECNT] = .DISPTR[DBG$W_DISP_LINECNT] + 1;
;  7091     7173  2      IF .DISPTR[DBG$W_DISP_LINECNT] EQL 1
;  7092     7174  2      THEN
;  7093     7175  2          DISPTR[DBG$L_DISP_WINDOW_PTR] = .DLEPTR;
;  7094     7176  2
;  7095     7177  2
```

```
;  7096          7178  2     ! If we are supposed to mark changed lines for this display, compare the
;  7097          7179  2     ! new text line to the first line on the old-text list.  If they differ,
;  7098          7180  2     ! we highlight the new text line with reverse video and bolding.
;  7099          7181  2     !
;  7100          7182  2     IF .DISPTR[DBG$V_DISP_MARKFLG] AND (NOT .DISPTR[DBG$V_DISP_NEWDISP])
;  7101          7183  2     THEN
;  7102          7184  3         BEGIN
;  7103          7185  3         CHANGE_FLAG = FALSE;
;  7104          7186  3         IF .DISPTR[DBG$L_DISP_OLDTXT_PTR] EQL 0
;  7105          7187  3         THEN
;  7106          7188  3             CHANGE_FLAG = TRUE
;  7107          7189  3
;  7108          7190  3         ELSE
;  7109          7191  4             BEGIN
;  7110          7192  4             OLDPTR = .DISPTR[DBG$L_DISP_OLDTXT_PTR];
;  7111          7193  4             DISPTR[DBG$L_DISP_OLDTXT_PTR] = .OLDPTR[DBG$L_DLINE_FLINK];
;  7112          7194  4             OLDTXT = OLDPTR[DBG$A_DLINE_TEXT];
;  7113          7195  4             CHANGE_FLAG = CH$NEQ(.TLEN, TEMPBUF, .OLDTXT[0], OLDTXT[1], 0);
;  7114          7196  4             DBG$REL_MEMORY(.OLDPTR);
;  7115          7197  3             END;
;  7116          7198  3
;  7117          7199  3         IF .CHANGE_FLAG
;  7118          7200  3         THEN
;  7119          7201  3             DLEPTR[DBG$B_DLINE_REND] = .DLEPTR[DBG$B_DLINE_REND] XOR
;  7120          7202  3                                 (DBG$M_DISP_REND_RV OR DBG$M_DISP_REND_BLD);
;  7121          7203  2         END;
;  7122          7204  2
;  7123          7205  2
;  7124          7206  2     ! If the screen window is not now positioned at the bottom of the display
;  7125          7207  2     ! text, force it to the bottom of the text.
;  7126          7208  2     !
;  7127          7209  2     NEW_DROW = .DISPTR[DBG$W_DISP_LINECNT] - .DISPTR[DBG$W_DISP_RLEN] + 1;
;  7128          7210  2     IF .DISPTR[DBG$W_DISP_DROW] LSS .NEW_DROW
;  7129          7211  2     THEN
;  7130          7212  3         BEGIN
;  7131          7213  3         OLD_DROW = .DISPTR[DBG$W_DISP_DROW];
;  7132          7214  3         DISPTR[DBG$W_DISP_DROW] = .NEW_DROW;
;  7133          7215  3         WSPTR = .DISPTR[DBG$L_DISP_END_LINE_PTR];
;  7134          7216  3         INCR I FROM 2 TO .DISPTR[DBG$W_DISP_RLEN] DO
;  7135          7217  3             WSPTR = .WSPTR[DBG$L_DLINE_BLINK];
;  7136          7218  3
;  7137          7219  3         DISPTR[DBG$L_DISP_WINDOW_PTR] = .WSPTR;
;  7138          7220  3         DISPTR[DBG$W_DISP_SCROLL] =
;  7139          7221  3                         .DISPTR[DBG$W_DISP_SCROLL] + .NEW_DROW - .OLD_DROW;
;  7140          7222  2         END;
;  7141          7223  2
;  7142          7224  2
;  7143          7225  2     ! The new text line has been written to the desired display.  Now return.
;  7144          7226  2     !
;  7145          7227  2     RETURN;
;  7146          7228  2
;  7147          7229  1     END;


                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0
```

```
54 49 52 57 5C 4E 45 45 52 43 53 47 42 44 17 007F4 P.AIA:  .ASCII  <23>\DBGSCREEN\<92>\WRITE_LINE 10\
                  30 31 20 45 4E 49 4C 5F 45 00803
54 49 52 57 5C 4E 45 45 52 43 53 47 42 44 17 0080C P.AIB:  .ASCII  <23>\DBGSCREEN\<92>\WRITE_LINE 20\
                  30 32 20 45 4E 49 4C 5F 45 0081B

                                                    .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        OFFC 00000          .ENTRY  DBG$SCR_WRITE_LINE, Save R2,R3,R4,R5,R6,R7,-; 7007
                                                                    R8,R9,RT0,R11
                                5E      FF00   CE  9E 00002         MOVAB   -256(SP), SP
                                         04    AC  D5 00007         TSTL    LENGTH                                7057
                                               15  18 0000A         BGEQ    1$
                                   00000000'   EF  9F 0000C         PUSHAB  P.AIA
                                               01  DD 00012         PUSHL   #1
                                   00028362    8F  DD 00014         PUSHL   #164706
                        00000000G  00          03  FB 0001A         CALLS   #3, LIB$SIGNAL
                                57        0C   AC  D0 00021 1$:     MOVL    DISPTR, R7                            7058
                                               15  12 00025         BNEQ    2$
                                   00000000'   EF  9F 00027         PUSHAB  P.AIB
                                               01  DD 0002D         PUSHL   #1
                                   00028362    8F  DD 0002F         PUSHL   #164706
                        00000000G  00          03  FB 00035         CALLS   #3, LIB$SIGNAL
                                               56  D4 0003C 2$:     CLRL    TLEN                                  7065
                                50             01  CE 0003E         MNEGL   #1, I                                 7072
                                               3B  11 00041         BRB     8$
                                09       08 BC40   91 00043 3$:     CMPB    @TEXTPTR[I], #9
                                               25  12 00048         BNEQ    6$
                     7E         00             56  7A 0004A         EMUL    #1, TLEN, #0, -(SP)                   7075
                     51         51             8E  7B 0004F         EDIV    #8, (SP)+, R1, R1
                                51             08  C3 00054         SUBL3   R1, #8, R1
                                               52  D4 00058         CLRL    J
                                               0D  11 0005A         BRB     5$
                                     864E      20  90 0005C 4$:     MOVB    #32, TEMPBUF[SP]                      7077
                                000000FF  8F     56  D1 00060         CMPL    TLEN, #255                            7079
                                               0C  18 00067         BGEQ    7$
                                EF             52  F3 00069 5$:     AOBLEQ  R1, J, 4$                             7075
                                               06  11 0006D         BRB     7$                                    7072
                                     864E      08 BC40   90 0006F 6$:     MOVB    @TEXTPTR[I], TEMPBUF[SP]              7089
                                000000FF  8F     56  D1 00075 7$:     CMPL    TLEN, #255                            7093
                                               05  18 0007C         BGEQ    9$
                                C0             50  04   AC  F2 0007E 8$:     AOBLSS  LENGTH, I, 3$                 7066
                                               58  D4 00083 9$:     CLRL    DLEPTR                                7102
                                5A       1C    A7  9E 00085         MOVAB   28(R7), R10                           7103
                                6A       02    AA  B1 00089         CMPW    2(R10), (R10)
                                               41  12 0008D         BNEQ    13$
                                58       20    A7  D0 0008F         MOVL    32(R7), DLEPTR                        7111
                                59             68  D0 00093         MOVL    (DLEPTR), FLINK                       7112
                                5B       04    A8  D0 00096         MOVL    4(DLEPTR), BLINK                      7113
                           04   A9             5B  D0 0009A         MOVL    BLINK, 4(FLINK)                       7114
                                6B             59  D0 0009E         MOVL    FLINK, (BLINK)                        7115
                                         02    AA  B7 000A1         DECW    2(R10)                                7116
                                01       18    A7  B1 000A4         CMPW    24(R7), #1                            7122
                                               05  1B 000A8         BLEQU   10$
                                         18    A7  B7 000AA         DECW    24(R7)                                7124
                                               0E  11 000AD         BRB     12$
```

```
                                                02 AA B5 000AF 10$:     TSTW    2(R10)                                  ; 7128
                                                02 12 000B2             BNEQ    11$                                     ;
                                                59 D4 000B4             CLRL    FLINK                                   ;
                                    28 A7       59 D0 000B6 11$:        MOVL    FLINK, 40(R7)                           ; 7129
                                                0C A7 B6 000BA          INCW    12(R7)                                  ; 7130
                56      0A  A8          08      00 ED 000BD 12$:        CMPZV   #0, #8, 10(DLEPTR), TLEN                ; 7138
                                                0B 18 000C3             BGEQ    13$                                     ;
                                                58 DD 000C5             PUSHL   DLEPTR                                  ; 7141
                                00000000G   00  01 FB 000C7             CALLS   #1, DBG$REL_MEMORY                      ;
                                                58 D4 000CE             CLRL    DLEPTR                                  ; 7142
                                                58 D5 000D0 13$:        TSTL    DLEPTR                                  ; 7151
                                                15 12 000D2             BNEQ    14$                                     ;
                                50              56      04 C7 000D4             DIVL3   #4, TLEN, R0                    ; 7154
                                                04 A0 9F 000D8          PUSHAB  4(R0)                                   ;
                                00000000G   00  01 FB 000DB             CALLS   #1, DBG$GET_MEMORY                      ;
                                            58  50 D0 000E2             MOVL    R0, DLEPTR                              ;
                                        0A  A8  56 90 000E5             MOVB    TLEN, 10(DLEPTR)                        ; 7155
                                        08  A8  09 A7 90 000E9 14$:     MOVB    9(R7), 8(DLEPTR)                        ; 7162
                                            50  0C A8 9E 000EE          MOVAB   12(R8), DLINE_TEXT                      ; 7163
                                            60      56 90 000F2         MOVB    TLEN, (DLINE_TEXT)                      ; 7164
                        01  A0              6E      56 28 000F5         MOVC3   TLEN, TEMPBUF, 1(DLINE_TEXT)            ; 7165
                                            59  20 A7 9E 000FA          MOVAB   32(R7), FLINK                           ; 7166
                                            5B  24 A7 D0 000FE          MOVL    36(R7), BLINK                           ; 7167
                                            68      59 D0 00102         MOVL    FLINK, (DLEPTR)                         ; 7168
                                        04  A8      5B D0 00105         MOVL    BLINK, 4(DLEPTR)                        ; 7169
                                        04  A9      58 D0 00109         MOVL    DLEPTR, 4(FLINK)                        ; 7170
                                            6B      58 D0 0010D         MOVL    DLEPTR, (BLINK)                         ; 7171
                                                02 AA B6 00110          INCW    2(R10)                                  ; 7172
                                            01  02 AA B1 00113         CMPW    2(R10), #1                              ; 7173
                                                04 12 00117             BNEQ    15$                                     ;
                                    28  A7          58 D0 00119         MOVL    DLEPTR, 40(R7)                          ; 7175
                            40      0A  A7      04 E1 0011D 15$:        BBC     #4, 10(R7), 19$                         ; 7182
                            3B      0A  A7      05 E0 00122             BBS     #5, 10(R7), 19$                         ;
                                                59 D4 00127             CLRL    CHANGE_FLAG                             ; 7185
                                                30 A7 D5 00129          TSTL    48(R7)                                  ; 7186
                                                05 12 0012C             BNEQ    16$                                     ;
                                            59      01 D0 0012E         MOVL    #1, CHANGE_FLAG                         ; 7188
                                                28 11 00131             BRB     18$                                     ;
                                            54  30 A7 D0 00133 16$:     MOVL    48(R7), OLDPTR                          ; 7192
                                        30  A7      64 D0 00137         MOVL    (OLDPTR), 48(R7)                        ; 7193
                                            50  0C A4 9E 0013B          MOVAB   12(R4), OLDTXT                          ; 7194
                                            51      60 9A 0013F         MOVZBL  (OLDTXT), R1                            ; 7195
                                                55 D4 00142             CLRL    R5                                      ;
        51              00                  6E      56 2D 00144         CMPC5   TLEN, TEMPBUF, #0, R1, 1(OLDTXT)        ;
                                                01 A0   00149           ;                                               ;
                                                02 13 0014B             BEQL    17$                                     ;
                                                55 D6 0014D             INCL    R5                                      ;
                                            59      55 D0 0014F 17$:    MOVL    R5, CHANGE_FLAG                         ;
                                                54 DD 00152             PUSHL   OLDPTR                                  ; 7196
                                00000000G   00  01 FB 00154             CALLS   #1, DBG$REL_MEMORY                      ;
                                            04  59 E9 0015B 18$:        BLBC    CHANGE_FLAG, 19$                        ; 7199
                                        08  A8      03 8C 0015E         XORB2   #3, 8(DLEPTR)                           ; 7202
                                            50  02 AA 3C 00162 19$:     MOVZWL  2(R10), R0                              ; 7209
                                            51  12 A7 3C 00166         MOVZWL  18(R7), R1                              ;
                                            50      51 C2 0016A         SUBL2   R1, R0                                  ;
                                                50 D6 0016D             INCL    NEW_DROW                                ;
        50      18  A7                      10      00 ED 0016F         CMPZV   #0, #16, 24(R7), NEW_DROW               ; 7210
```

```
                                                2D  18 00175          BGEQ    22$                                                 ;
                                54      18      A7  3C 00177          MOVZWL  24(R7), OLD_DROW                                    ; 7213
                            18  A7              50  B0 0017B          MOVW    NEW_DROW, 24(R7)                                    ; 7214
                                51      24      A7  D0 0017F          MOVL    36(R7), WSPTR                                       ; 7215
                                53      12      A7  3C 00183          MOVZWL  18(R7), R3                                          ; 7216
                                52              01  D0 00187          MOVL    #1, I                                               ;
                                                04  11 0018A          BRB     21$                                                 ;
                                51      04      A1  D0 0018C 20$:     MOVL    4(WSPTR), WSPTR                                     ; 7217
                        F8      52              53  F3 00190 21$:     AOBLEQ  R3, I, 20$                                          ;
                            28  A7              51  D0 00194          MOVL    WSPTR, 40(R7)                                       ; 7219
                                51      0C      A7  32 00198          CVTWL   12(R7), R1                                          ; 7221
                                50              51  C0 0019C          ADDL2   R1, R0                                              ;
                    0C      A7  50              54  A3 0019F          SUBW3   OLD_DROW, R0, 12(R7)                                ;
                                                04  001A4 22$:        RET                                                         ; 7229

; Routine Size:  421 bytes,    Routine Base:  DBG$CODE + 3B8E
```

```
;  7149          7230  1 ROUTINE FORMAT_SOURCE_LINE(DISPTR, DLEPTR, TEXTPTR, TEXTCNT, BUFFER): NOVALUE =
;  7150          7231  1 !
;  7151          7232  1 ! FUNCTION
;  7152          7233  1 !       This routine formats a source line for output as part of a Screen
;  7153          7234  1 !       Display.  Source lines are formatted differently than other lines
;  7154          7235  1 !       for two reasons: first, the Display Line Entry has a different format
;  7155          7236  1 !       for source lines, and second, scrolling to the right or left is done
;  7156          7237  1 !       in such a way that the line number cannot be scrolled off the left
;  7157          7238  1 !       end of the display.  In addition, the "mark line" of a source display
;  7158          7239  1 !       is marked with a small pointer "->", which does not happen with non-
;  7159          7240  1 !       source lines.
;  7160          7241  1 !
;  7161          7242  1 ! INPUTS
;  7162          7243  1 !       DISPTR  - A pointer to the Screen Display Entry for the display that
;  7163          7244  1 !                 contains the line being formatted.
;  7164          7245  1 !
;  7165          7246  1 !       DLEPTR  - A pointer to the Display Line Entry for the source line
;  7166          7247  1 !                 being formatted.
;  7167          7248  1 !
;  7168          7249  1 !       TEXTPTR - The address of a longword location to receive a pointer to
;  7169          7250  1 !                 formatted source line.
;  7170          7251  1 !
;  7171          7252  1 !       TEXTCNT - The address of a longword location to receive the length
;  7172          7253  1 !                 of the formatted source line.
;  7173          7254  1 !
;  7174          7255  1 !       BUFFER  - A pointer to a 132-character buffer to be used for formatting
;  7175          7256  1 !                 the source line.
;  7176          7257  1 !
;  7177          7258  1 ! OUTPUTS
;  7178          7259  1 !       TEXTPTR - The address of the formatted source line is returned to the
;  7179          7260  1 !                 TEXTPTR location.
;  7180          7261  1 !
;  7181          7262  1 !       TEXTCNT - The length of the formatted source line is returned to the
;  7182          7263  1 !                 TEXTCNT location.
;  7183          7264  1 !
;  7184          7265  1
;  7185          7266  2     BEGIN
;  7186          7267  2
;  7187          7268  2     MAP
;  7188          7269  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to source Display Entry
;  7189          7270  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Display Line Entry
;  7190          7271  2         TEXTPTR: REF VECTOR[1,LONG],    ! Pointer to returned text pointer
;  7191          7272  2         TEXTCNT: REF VECTOR[1,LONG],    ! Pointer to returned text length
;  7192          7273  2         BUFFER: REF VECTOR[,BYTE];      ! Pointer to editing buffer
;  7193          7274  2
;  7194          7275  2     LOCAL
;  7195          7276  2         LINENUM,                        ! Line number of this source line
;  7196          7277  2         TLEN,                           ! Length of edited source line
;  7197          7278  2         TPTR: REF VECTOR[,BYTE];        ! Pointer to edited source line
;  7198          7279  2
;  7199          7280  2
;  7200          7281  2
;  7201          7282  2     ! Edit the source line number of this source line into the editing buffer.
;  7202          7283  2     ! The number is truncated to six digits.  Note that we include a "->" in
;  7203          7284  2     ! front of the line number if this is the "marked" line of the display.
;  7204          7285  2     !
;  7205          7286  2     CH$FILL(' ', 8, .BUFFER);
```

```
;  7206          7287  2      BUFFER[6] = ':';
;  7207          7288  2      IF .D'EPTR[DBG$L_DLINE_LINUM] GTR 0
;  7208          7289  2      THEN
;  7209          7290  3          BEGIN
;  7210          7291  3          LINENUM = .DLEPTR[DBG$L_DLINE_LINUM];
;  7211          7292  3          DECR I FROM 5 TO 0 DO
;  7212          7293  4              BEGIN
;  7213          7294  4              BUFFER[.I] = (.LINENUM MOD 10) + '0';
;  7214          7295  4              LINENUM = .LINENUM/10;
;  7215          7296  4              IF .LINENUM EQL 0 THEN EXITLOOP;
;  7216          7297  3              END;
;  7217          7298  3
;  7218          7299  3          IF .DLEPTR[DBG$L_DLINE_LINUM] EQL .DISPTR[DBG$L_DISP_MARKLINE]
;  7219          7300  3          THEN
;  7220          7301  4              BEGIN
;  7221          7302  4              BUFFER[0] = '-';
;  7222          7303  4              BUFFER[1] = '>';
;  7223          7304  3              END;
;  7224          7305  3
;  7225          7306  2          END;
;  7226          7307  2
;  7227          7308  2
;  7228          7309  2      ! Copy the actual text of the source line into BUFFER.  The right scrolling
;  7229          7310  2      ! specified in DCOL is performed at this time.
;  7230          7311  2      !
;  7231          7312  2      TPTR = DLEPTR[DBG$A_DLINE_TEXT2];
;  7232          7313  2      TLEN = MIN(.DBG$SRC_TERM_WIDTH - 8,
;  7233          7314  2                 MAX(0, .TPTR[0] - .DISPTR[DBG$W_DISP_DCOL] + 1));
;  7234          7315  2      CH$MOVE(.TLEN, TPTR[.DISPTR[DBG$W_DISP_DCOL]], BUFFER[8]);
;  7235          7316  2
;  7236          7317  2
;  7237          7318  2      ! Finally return the length and location of the edited source line and
;  7238          7319  2      ! return control to the caller.
;  7239          7320  2      !
;  7240          7321  2      TEXTCNT[0] = .TLEN + 8;
;  7241          7322  2      TEXTPTR[0] = .BUFFER;
;  7242          7323  2      RETURN;
;  7243          7324  2
;  7244          7325  1      END;


                                        00FC 00000 FORMAT_SOURCE_LINE:
                                                   .WORD   Save R2,R3,R4,R5,R6,R7                  ;  7230
                          56     14     AC  D0 00002       MOVL    BUFFER, R6                      ;  7286
08                20      6E            00  2C 00006       MOVC5   #0, (SP), #32, #8, (R6)         ;
                                        66     0000B       
                    06    A6            3A  90 0000C       MOVB    #58, 6(R6)                      ;  7287
                          51     08     AC  D0 00010       MOVL    DLEPTR, R1                      ;  7288
                          52     10     A1  D0 00014       MOVL    16(R1), R2                      ;
                                        2C  15 00018       BLEQ    3$                              ;
                          54            52  D0 0001A       MOVL    R2, LINENUM                     ;  7291
                          53            05  D0 0001D       MOVL    #5, I                           ;  7292
7E                00      54            01  7A 00020 1$:   EMUL    #1, LINENUM, #0, -(SP)          ;  7294
50                50      8E            0A  7B 00025       EDIV    #10, (SP)+, R0, R0              ;
```

```
                6346            50              30  81 0002A            ADDB3   #48, R0, (I)[R6]                        ;
                                54              0A  C6 0002F            DIVL2   #10, LINENUM                    ; 7295
                                                03  13 00032            BEQL    2$                              ; 7296
                                E9              53  F4 00034            SOBGEQ  I, 1$                           ; 7292
                                50      04      AC  D0 00037 2$:        MOVL    DISPTR, R0                      ; 7299
                        3C      A0              52  D1 0003B            CMPL    R2, 60(R0)                      ;
                                                05  12 0003F            BNEQ    3$                              ;
                                66      3E2D    8F  B0 00041            MOVW    #15917, (R6)                    ; 7302
                                50      14      A1  9E 00046 3$:        MOVAB   20(R1), TPTR                    ; 7312
                    52 00000000G        00      08  C3 0004A            SUBL3   #8, DBG$SRC_TERM_WIDTH, R2      ; 7313
                                51      04      AC  D0 00052            MOVL    DISPTR, R1                      ; 7314
                                53      1A      A1  3C 00056            MOVZWL  26(R1), R3                      ;
                                51              60  9A 0005A            MOVZBL  (TPTR), R1                      ;
                                51              53  C2 0005D            SUBL2   R3, R1                          ;
                                                51  D6 00060            INCL    R1                              ;
                                                02  18 00062            BGEQ    4$                              ;
                                                51  D4 00064            CLRL    R1                              ;
                                51              52  D1 00066 4$:        CMPL    R2, R1                          ;
                                                03  15 00069            BLEQ    5$                              ;
                                52              51  D0 0006B            MOVL    R1, R2                          ;
                                57              52  D0 0006E 5$:        MOVL    R2, TLEN                        ; 7313
            08  A6                  6340        57  28 00071            MOVC3   TLEN, (R3)[TPTR], 8(R6)         ; 7315
                        10  BC      08  A7  9E 00077            MOVAB   8(R7), @TEXTCNT                 ; 7321
                        0C  BC          56  D0 0007C            MOVL    R6, @TEXTPTR                    ; 7322
                                            04 00080            RET                                     ; 7325

; Routine Size:  129 bytes,    Routine Base:  DBG$CODE + 3D33
```

```
;  7246     7326  1 ROUTINE HANDLER_EXECUTE_SAVE(SIGARG, MECHARG, ENBLARG) =
;  7247     7327  1 !
;  7248     7328  1 ! FUNCTION
;  7249     7329  1 !       This routine is the Condition Handler for the DBG$SCR_EXECUTE_SAVE_CMD
;  7250     7330  1 !       routine.  Its purpose is to release the Screen Display Entry and its
;  7251     7331  1 !       associated Display Line Entries in case the DEBUG Memory Pool runs out
;  7252     7332  1 !       of memory in the middle of building a new Display Entry while executing
;  7253     7333  1 !       a SAVE command.  After releasing all such memory (if any) in response
;  7254     7334  1 !       to the DBG$_NOFREE condition, this routine resignals the condition.
;  7255     7335  1 !       Any other condition is resignalled immediately.
;  7256     7336  1 !
;  7257     7337  1 ! INPUTS
;  7258     7338  1 !       SIGARG  - The condition Signal Argument vector.
;  7259     7339  1 !
;  7260     7340  1 !       MECHARG - The condition Mechanism Argument vector.
;  7261     7341  1 !
;  7262     7342  1 !       ENBLARG - The condition Enable Argument vector.  This handler expects
;  7263     7343  1 !                 this vector to contain one element, ENBLARG[1], which points
;  7264     7344  1 !                 to the Screen Display Entry to be released in case the "no
;  7265     7345  1 !                 free storage" error occurred.
;  7266     7346  1 !
;  7267     7347  1 ! OUTPUTS
;  7268     7348  1 !       This routine always returns SS$_RESIGNAL as its result.
;  7269     7349  1 !
;  7270     7350  1
;  7271     7351  2     BEGIN
;  7272     7352  2
;  7273     7353  2     MAP
;  7274     7354  2         SIGARG: REF VECTOR[,LONG],      ! Pointer to Signal Argument vector
;  7275     7355  2         ENBLARG: REF VECTOR[,LONG];     ! Pointer to Enable Argument vector
;  7276     7356  2
;  7277     7357  2     LOCAL
;  7278     7358  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to Display Entry to release
;  7279     7359  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to Display Line Entry
;  7280     7360  2         NEXTDLE;                        ! Pointer to next Display Line Entry
;  7281     7361  2
;  7282     7362  2
;  7283     7363  2
;  7284     7364  2     ! Unless this is the DBG$_NOFREE error condition, we resignal immediately.
;  7285     7365  2     ! No other condition should occur, so we want to preserve the current state
;  7286     7366  2     ! for debugging if something else did occur.  We also resignal immediately
;  7287     7367  2     ! if there is no Screen Display Entry to release.
;  7288     7368  2     !
;  7289     7369  2     IF .SIGARG[1] NEQ DBG$_NOFREE THEN RETURN SS$_RESIGNAL;
;  7290     7370  2     IF .ENBLARG[1] EQL 0 THEN RETURN SS$_RESIGNAL;
;  7291     7371  2
;  7292     7372  2
;  7293     7373  2     ! There is a Screen Display Entry to release.  First loop to release all
;  7294     7374  2     ! Display Line Entries attached to the Display Entry.
;  7295     7375  2     !
;  7296     7376  2     DISPTR = .ENBLARG[1];
;  7297     7377  2     DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  7298     7378  2     WHILE .DLEPTR NEQ DISPTR[DBG$L_DISP_START_LINE_PTR] DO
;  7299     7379  3         BEGIN
;  7300     7380  3         NEXTDLE = .DLEPTR[DBG$L_DLINE_FLINK];
;  7301     7381  3         DBG$REL_MEMORY(.DLEPTR);
;  7302     7382  3         DLEPTR = .NEXTDLE;
```

```
;  7303          7383  2          END;
;  7304          7384  2
;  7305          7385  2
;  7306          7386  2      ! Then release the Screen Display Entry itself and resignal the condition.
;  7307          7387  2      !
;  7308          7388  2      DBG$REL_MEMORY(.DISPTR);
;  7309          7389  2      RETURN SS$_RESIGNAL;
;  7310          7390  2
;  7311          7391  1      END;


                                        003C 00000 HANDLER_EXECUTE_SAVE:
                                                   .WORD   Save R2,R3,R4,R5                        ;  7326
                    55 00000000G    00  9E 00002           MOVAB   DBG$REL_MEMORY, R5
                    50         04   AC  D0 00009           MOVL    SIGARG, R0                      ;  7369
            00028332   8F         04   A0  D1 0000D           CMPL    4(R0), #164658
                                   2C  12 00015           BNEQ    3$
                    50         0C   AC  D0 00017           MOVL    ENBLARG, R0                     ;  7370
                               04   A0  D5 0001B           TSTL    4(R0)
                                   23  13 0001E           BEQL    3$
                    53         04   A0  D0 00020           MOVL    4(R0), DISPTR                   ;  7376
                    52         20   A3  D0 00024           MOVL    32(DISPTR), DLEPTR              ;  7377
                    50         20   A3  9E 00028 1$:       MOVAB   32(DISPTR), R0                  ;  7378
                    50              52  D1 0002C           CMPL    DLEPTR, R0
                                   0D  13 0002F           BEQL    2$
                    54              62  D0 00031           MOVL    (DLEPTR), NEXTDLE               ;  7380
                                   52  DD 00034           PUSHL   DLEPTR                          ;  7381
                    65              01  FB 00036           CALLS   #1, DBG$REL_MEMORY
                    52              54  D0 00039           MOVL    NEXTDLE, DLEPTR                 ;  7382
                                   EA  11 0003C           BRB     1$                              ;  7378
                                   53  DD 0003E 2$:       PUSHL   DISPTR                          ;  7388
                    65              01  FB 00040           CALLS   #1, DBG$REL_MEMORY
                    50      0918    8F  3C 00043 3$:       MOVZWL  #2328, R0                       ;  7389
                                       04 00048           RET                                     ;  7391

; Routine Size:  73 bytes,    Routine Base:  DBG$CODE + 3DB4
```

```
;                    7392  1 ROUTINE HANDLER_SCROLL_SOURCE(SIGARG, MECHARG, ENBLARG) =
;                    7393  1 !
;                    7394  1 ! FUNCTION
;                    7395  1 !       This routine is the Condition Handler for the DBG$SCR_SCROLL_SOURCE_xx
;                    7396  1 !       routines.  Its purpose is to restore the value of DBG$GL_SCREEN_SOURCE
;                    7397  1 !       in case an error condition forces a premature exit from the current
;                    7398  1 !       source display scrolling operation.  The address of the value to be
;                    7399  1 !       restored is passed in as an Enable argument.
;                    7400  1 !
;                    7401  1 ! INPUTS
;                    7402  1 !       SIGARG  - The condition Signal Argument vector.
;                    7403  1 !
;                    7404  1 !       MECHARG - The condition Mechanism Argument vector.
;                    7405  1 !
;                    7406  1 !       ENBLARG - The condition Enable Argument vector.  This handler expects
;                    7407  1 !                 this vector to contain one element, ENBLARG[1], which con-
;                    7408  1 !                 tains the the address of the location that contains the
;                    7409  1 !                 value to which DBG$GL_SCREEN_SOURCE should be restored.
;                    7410  1 !
;                    7411  1 ! OUTPUTS
;                    7412  1 !       This routine always returns SS$_RESIGNAL as its result.
;                    7413  1 !
;                    7414  1
;                    7415  2     BEGIN
;                    7416  2
;                    7417  2     MAP
;                    7418  2         ENBLARG: REF VECTOR[,LONG];     ! Pointer to Enable Argument vector
;                    7419  2
;                    7420  2     LOCAL
;                    7421  2         PTR: REF VECTOR[1, LONG];       ! Pointer to value to be restored
;                    7422  2
;                    7423  2
;                    7424  2
;                    7425  2     ! Restore the value of DBG$GL_SCREEN_SOURCE and resignal the condition.
;                    7426  2     !
;                    7427  2     PTR = .ENBLARG[1];
;                    7428  2     DBG$GL_SCREEN_SOURCE = .PTR[0];
;                    7429  2     RETURN SS$_RESIGNAL;
;                    7430  2
;                    7431  1     END;


                                        0000 00000 HANDLER_SCROLL_SOURCE:
                                                   .WORD   Save nothing                          ; 7392
                          50       0C   AC  D0 000C2      MOVL    ENBLARG, R0                   ; 7427
                          50       04   A0  D0 00006      MOVL    4(R0), PTR                    ;
00000000'                 EF            60  D0 0000A      MOVL    (PTR), DBG$GL_SCREEN_SOURCE   ; 7428
                          50     0918   8F  3C 00011      MOVZWL  #2328, R0                     ; 7429
                                        04 00016          RET                                   ; 7431

; Routine Size:  23 bytes,    Routine Base:  DBG$CODE + 3DFD
```

```
;  7354     7432  1 ROUTINE PARSE_DISPLAY_NAME(INPUT_DESC, PERCENT_ALLOWED) =
;  7355     7433  1 !
;  7356     7434  1 ! FUNCTION
;  7357     7435  1 !       This routine parses (scans and picks up) a screen display name.  It
;  7358     7436  1 !       is called during the parsing of all Screen Debugging commands which
;  7359     7437  1 !       have display names in their syntax.
;  7360     7438  1 !
;  7361     7439  1 ! INPUTS
;  7362     7440  1 !       INPUT_DESC - A string descriptor to the input line being parsed.
;  7363     7441  1 !                    The descriptor is assumed to point to the start of the
;  7364     7442  1 !                    display name or to a blank or tab before it.
;  7365     7443  1 !
;  7366     7444  1 !       PERCENT_ALLOWED - If this flag is TRUE, the display name is allowed
;  7367     7445  1 !                    to begin with a percent sign, thus denoting one of the
;  7368     7446  1 !                    display pseudo-symbols such as %NEXTDISP, etc.  If this
;  7369     7447  1 !                    flag is FALSE, the percent sign is not allowed to start
;  7370     7448  1 !                    the display name.
;  7371     7449  1 !
;  7372     7450  1 ! OUTPUTS
;  7373     7451  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  7374     7452  1 !                    first character after the display name picked up by this
;  7375     7453  1 !                    routine.
;  7376     7454  1 !
;  7377     7455  1 !       A pointer to the display name in Counted ASCII format is returned as
;  7378     7456  1 !                    the routine's value.
;  7379     7457  1 !
;  7380     7458  1
;  7381     7459  2     BEGIN
;  7382     7460  2
;  7383     7461  2     MAP
;  7384     7462  2         INPUT_DESC: REF BLOCK[,BYTE];   ! Pointer to input string descriptor
;  7385     7463  2
;  7386     7464  2     LOCAL
;  7387     7465  2         CHARLEN,                        ! Character length of input string
;  7388     7466  2         CHARPTR: REF VECTOR[,BYTE],     ! Character pointer into input string
;  7389     7467  2         NAMEPTR: REF VECTOR[,BYTE];     ! Pointer to the ASCIC display name
;  7390     7468  2
;  7391     7469  2
;  7392     7470  2
;  7393     7471  2     ! Set up the character pointer and string length we will use and update
;  7394     7472  2     ! as the display name is scanned.  Also get a buffer to hold the name.
;  7395     7473  2     !
;  7396     7474  2     CHARPTR = .INPUT_DESC[DSC$A_POINTER];
;  7397     7475  2     CHARLEN = .INPUT_DESC[DSC$W_LENGTH];
;  7398     7476  2     NAMEPTR = DBG$GET_TEMPMEM(84/4);
;  7399     7477  2     NAMEPTR[0] = 0;
;  7400     7478  2
;  7401     7479  2
;  7402     7480  2     ! First scan past any leading blanks or tabs.
;  7403     7481  2     !
;  7404     7482  2     WHILE (.CHARPTR[0] EQL ' ') OR (.CHARPTR[0] EQL TAB_CHAR) DO
;  7405     7483  3         BEGIN
;  7406     7484  3         CHARPTR = .CHARPTR + 1;
;  7407     7485  3         CHARLEN = .CHARLEN - 1;
;  7408     7486  2         END;
;  7409     7487  2
;  7410     7488  2
```

```
;  7411          7489  2     ! Then scan past the leading percent sign if there is one and if one is
;  7412          7490  2     ! allowed on this command.
;  7413          7491  2     !
;  7414          7492  2     IF .PERCENT_ALLOWED AND (.CHARPTR[0] EQL '%')
;  7415          7493  2     THEN
;  7416          7494  3         BEGIN
;  7417          7495  3         CHARPTR = .CHARPTR + 1;
;  7418          7496  3         CHARLEN = .CHARLEN - 1;
;  7419          7497  3         NAMEPTR[0] = 1;
;  7420          7498  3         NAMEPTR[1] = '%';
;  7421          7499  3         END;
;  7422          7500  2
;  7423          7501  2
;  7424          7502  2     ! Now scan the display name.  Copy the name itself into the NAMEPTR buffer.
;  7425          7503  2     !
;  7426          7504  2     WHILE ((.CHARPTR[0] GEQ 'A') AND (.CHARPTR[0] LEQ 'Z')) OR
;  7427          7505  2           ((.CHARPTR[0] GEQ '0') AND (.CHARPTR[0] LEQ '9')) OR
;  7428          7506  3           (.CHARPTR[0] EQL '$') OR (.CHARPTR[0] EQL '_')
;  7429          7507  2     DO
;  7430          7508  3         BEGIN
;  7431          7509  3         IF .NAMEPTR[0] GEQ 80 THEN SIGNAL(DBG$_NAMTOOLONG, 1, .NAMEPTR);
;  7432          7510  3         NAMEPTR[0] = .NAMEPTR[0] + 1;
;  7433          7511  3         NAMEPTR[.NAMEPTR[0]] = .CHARPTR[0];
;  7434          7512  3         CHARPTR = .CHARPTR + 1;
;  7435          7513  3         CHARLEN = .CHARLEN - 1;
;  7436          7514  3         END;
;  7437          7515  2
;  7438          7516  2
;  7439          7517  2     ! We are done scanning the display name.  Update the input string descriptor
;  7440          7518  2     ! to indicate the new parse location.  Then return the display name pointer.
;  7441          7519  2     !
;  7442          7520  2     IF .NAMEPTR[0] EQL 0 THEN DBG$SYNTAX_ERROR(.INPUT_DESC);
;  7443          7521  2     INPUT_DESC[DSC$A_POINTER] = .CHARPTR;
;  7444          7522  2     INPUT_DESC[DSC$W_LENGTH] = .CHARLEN;
;  7445          7523  2     RETURN .NAMEPTR;
;  7446          7524  2
;  7447          7525  1     END;


                                003C 00000 PARSE_DISPLAY_NAME:
                                           .WORD   Save R2,R3,R4,R5                ;  7432
                    54    04    AC  D0 00002       MOVL    INPUT_DESC, R4          ;  7474
                    53    04    A4  D0 00006       MOVL    4(R4), CHARPTR          ;
                    55          64  3C 0000A       MOVZWL  (R4), CHARLEN           ;  7475
                                15  DD 0000D       PUSHL   #21                     ;  7476
00000000G           00          01  FB 0000F       CALLS   #1, DBG$GET_TEMPMEM     ;
                    52          50  D0 00016       MOVL    R0, NAMEPTR             ;
                                62  94 00019       CLRB    (NAMEPTR)               ;  7477
                    20          63  91 0001B 1$:    CMPB    (CHARPTR), #32          ;  7482
                                05  13 0001E       BEQL    2$                      ;
                    09          63  91 00020       CMPB    (CHARPTR), #9           ;
                                06  12 00023       BNEQ    3$                      ;
                                53  D6 00025 2$:    INCL    CHARPTR                 ;  7484
                                55  D7 00027       DECL    CHARLEN                 ;  7485
```

```
                                F0  11 00029          BRB     1$                                      ;  7482
                          0E      08  AC  E9 0002B 3$:    BLBC    PERCENT_ALLOWED, 4$                     ;  7492
                          25          63  91 0002F          CMPB    (CHARPTR), #37                          ;
                                      09  12 00032          BNEQ    4$                                      ;
                                      53  D6 00034          INCL    CHARPTR                                 ;  7495
                                      55  D7 00036          DECL    CHARLEN                                 ;  7496
                          62      2501    8F  B0 00038          MOVW    #9473, (NAMEPTR)                        ;  7497
                    41    8F            63  91 0003D 4$:     CMPB    (CHARPTR), #65                          ;  7504
                                      06  1F 00041          BLSSU   5$                                      ;
                    5A    8F            63  91 00043          CMPB    (CHARPTR), #90                          ;
                                      15  1B 00047          BLEQU   7$                                      ;
                          30            63  91 00049 5$:     CMPB    (CHARPTR), #48                          ;  7505
                                      05  1F 0004C          BLSSU   6$                                      ;
                          39            63  91 0004E          CMPB    (CHARPTR), #57                          ;
                                      08  1B 00051          BLEQU   7$                                      ;
                          24            63  91 00053 6$:     CMPB    (CHARPTR), #36                          ;  7506
                                      06  13 00056          BEQL    7$                                      ;
                    5F    8F            63  91 00058          CMPB    (CHARPTR), #95                          ;
                                      24  12 0005C          BNEQ    9$                                      ;
                    50    8F            62  91 0005E 7$:     CMPB    (NAMEPTR), #80                          ;  7509
                                      11  1F 00062          BLSSU   8$                                      ;
                                      52  DD 00064          PUSHL   NAMEPTR                                 ;
                                      01  DD 00066          PUSHL   #1                                      ;
                          00028A82    8F  DD 00068          PUSHL   #166530                                 ;
          00000000G       00          03  FB 0006E          CALLS   #3, LIB$SIGNAL                          ;
                                      62  96 00075 8$:     INCB    (NAMEPTR)                               ;  7510
                              5C      62  9A 00077          MOVZBL  (NAMEPTR), R0                           ;  7511
                            6042      83  90 0007A          MOVB    (CHARPTR)+, (R0)[NAMEPTR]               ;
                                      55  D7 0007E          DECL    CHARLEN                                 ;  7513
                                      BB  11 00080          BRB     4$                                      ;  7504
                                      62  95 00082 9$:     TSTB    (NAMEPTR)                               ;  7520
                                      09  12 00084          BNEQ    10$                                     ;
                                      54  DD 00086          PUSHL   R4                                      ;
          00000000G       00          01  FB 00088          CALLS   #1, DBG$SYNTAX_ERROR                    ;
                    04    A4            53  D0 0008F 10$:    MOVL    CHARPTR, 4(R4)                          ;  7521
                              64        55  B0 00093          MOVW    CHARLEN, (R4)                           ;  7522
                              50        52  D0 00096          MOVL    NAMEPTR, R0                             ;  7523
                                        04  00099          RET                                             ;  7525

; Routine Size:  154 bytes,    Routine Base:  DBG$CODE + 3E14
```

```
;  7449          7526  1 ROUTINE PARSE_WINDOW_NAME(INPUT_DESC) =
;  7450          7527  1 !
;  7451          7528  1 ! FUNCTION
;  7452          7529  1 !       This routine parses (scans and picks up) a screen window name.  It
;  7453          7530  1 !       is called during the parsing of all Screen Debugging commands which
;  7454          7531  1 !       have window names in their syntax.
;  7455          7532  1 !
;  7456          7533  1 ! INPUTS
;  7457          7534  1 !       INPUT_DESC - A string descriptor to the input line being parsed.
;  7458          7535  1 !                    The descriptor is assumed to point to the start of the
;  7459          7536  1 !                    window name or to a blank or tab before it.
;  7460          7537  1 !
;  7461          7538  1 ! OUTPUTS
;  7462          7539  1 !       INPUT_DESC - The input string descriptor is updated to point to the
;  7463          7540  1 !                    first character after the window name picked up by this
;  7464          7541  1 !                    routine.
;  7465          7542  1 !
;  7466          7543  1 !       A pointer to the window name in Counted ASCII format is returned as
;  7467          7544  1 !                    the routine's value.
;  7468          7545  1 !
;  7469          7546  1
;  7470          7547  2     BEGIN
;  7471          7548  2
;  7472          7549  2     MAP
;  7473          7550  2         INPUT_DESC: REF BLOCK[,BYTE];   ! Pointer to input string descriptor
;  7474          7551  2
;  7475          7552  2     LOCAL
;  7476          7553  2         CHARLEN,                        ! Character length of input string
;  7477          7554  2         CHARPTR: REF VECTOR[,BYTE],     ! Character pointer into input string
;  7478          7555  2         NAMEPTR: REF VECTOR[,BYTE];     ! Pointer to the ASCIC window name
;  7479          7556  2
;  7480          7557  2
;  7481          7558  2
;  7482          7559  2     ! Set up the character pointer and string length we will use and update
;  7483          7560  2     ! as the window name is scanned.  Also get a buffer to hold the name.
;  7484          7561  2     !
;  7485          7562  2     CHARPTR = .INPUT_DESC[DSC$A_POINTER];
;  7486          7563  2     CHARLEN = .INPUT_DESC[DSC$W_LENGTH];
;  7487          7564  2     NAMEPTR = DBG$GET_TEMPMEM(84/4);
;  7488          7565  2     NAMEPTR[0] = 0;
;  7489          7566  2
;  7490          7567  2
;  7491          7568  2     ! First scan past any leading blanks or tabs.
;  7492          7569  2     !
;  7493          7570  2     WHILE (.CHARPTR[0] EQL ' ') OR (.CHARPTR[0] EQL TAB_CHAR) DO
;  7494          7571  3         BEGIN
;  7495          7572  3         CHARPTR = .CHARPTR + 1;
;  7496          7573  3         CHARLEN = .CHARLEN - 1;
;  7497          7574  2         END;
;  7498          7575  2
;  7499          7576  2
;  7500          7577  2     ! Now scan the window name.  Copy the name itself into the NAMEPTR buffer.
;  7501          7578  2     !
;  7502          7579  2     WHILE ((.CHARPTR[0] GEQ 'A') AND (.CHARPTR[0] LEQ 'Z')) OR
;  7503          7580  2           ((.CHARPTR[0] GEQ '0') AND (.CHARPTR[0] LEQ '9')) OR
;  7504          7581  3           (.CHARPTR[0] EQL '$') OR (.CHARPTR[0] EQL '_')
;  7505          7582  2     DO
```

```
;  7506          7583  3          BEGIN
;  7507          7584  3          IF .NAMEPTR[0] GEQ 80 THEN SIGNAL(DBG$_NAMTOOLONG, 1, .NAMEPTR);
;  7508          7585  3          NAMEPTR[0] = .NAMEPTR[0] + 1;
;  7509          7586  3          NAMEPTR[.NAMEPTR[0]] = .CHARPTR[0];
;  7510          7587  3          CHARPTR = .CHARPTR + 1;
;  7511          7588  3          CHARLEN = .CHARLEN - 1;
;  7512          7589  2          END;
;  7513          7590  2
;  7514          7591  2
;  7515          7592  2      ! We are done scanning the window name.  Update the input string descriptor
;  7516          7593  2      ! to indicate the new parse location.  Then return the window name pointer.
;  7517          7594  2      !
;  7518          7595  2      IF .NAMEPTR[0] EQL 0 THEN DBG$SYNTAX_ERROR(.INPUT_DESC);
;  7519          7596  2      INPUT_DESC[DSC$A_POINTER] = .CHARPTR;
;  7520          7597  2      INPUT_DESC[DSC$W_LENGTH] = .CHARLEN;
;  7521          7598  2      RETURN .NAMEPTR;
;  7522          7599  2
;  7523          7600  1      END;


                                        003C 00000 PARSE_WINDOW_NAME:
                                                   .WORD   Save R2,R3,R4,R5                        ;  7526
                          54        04   AC  D0 00002     MOVL    INPUT_DESC, R4                  ;  7562
                          53        04   A4  D0 00006     MOVL    4(R4), CHARPTR                  ;
                          55             64  3C 0000A     MOVZWL  (R4), CHARLEN                   ;  7563
                                         15  DD 0000D     PUSHL   #21                             ;  7564
                00000000G 00             01  FB 0000F     CALLS   #1, DBG$GET_TEMPMEM             ;
                          52             50  D0 00016     MOVL    R0, NAMEPTR                     ;
                                         62  94 00019     CLRB    (NAMEPTR)                       ;  7565
                          20             63  91 0001B 1$:  CMPB    (CHARPTR), #32                  ;  7570
                                         05  13 0001E     BEQL    2$                              ;
                          09             63  91 00020     CMPB    (CHARPTR), #9                   ;
                                         06  12 00023     BNEQ    3$                              ;
                                         53  D6 00025 2$:  INCL    CHARPTR                         ;  7572
                                         55  D7 00027     DECL    CHARLEN                         ;  7573
                                         F0  11 00029     BRB     1$                              ;  7570
                          41   8F        63  91 0002B 3$:  CMPB    (CHARPTR), #65                  ;  7579
                                         06  1F 0002F     BLSSU   4$                              ;
                          5A   8F        63  91 00031     CMPB    (CHARPTR), #90                  ;
                                         15  1B C0035     BLEQU   6$                              ;
                          30             63  91 00037 4$:  CMPB    (CHARPTR), #48                  ;  7580
                                         05  1F 0003A     BLSSU   5$                              ;
                          39             63  91 0003C     CMPB    (CHARPTR), #57                  ;
                                         0B  1B 0003F     BLEQU   6$                              ;
                          24             63  91 00041 5$:  CMPB    (CHARPTR), #36                  ;  7581
                                         06  13 00044     BEQL    6$                              ;
                          5F   8F        63  91 00046     CMPB    (CHARPTR), #95                  ;
                                         24  12 0004A     BNEQ    8$                              ;
                          50   8F        62  91 0004C 6$:  CMPB    (NAMEPTR), #80                  ;  7584
                                         11  1F 00050     BLSSU   7$                              ;
                                         52  DD 00052     PUSHL   NAMEPTR                         ;
                                         01  DD 00054     PUSHL   #1                              ;
                                   00028A82  8F  DD 00056 PUSHL   #166530                         ;
                00000000G 00             03  FB 0005C     CALLS   #3, LIB$SIGNAL                  ;
```

```
                                        62  96 00063 7$:    INCB    (NAMEPTR)                                       ; 7585
                                     50 62  9A 00065         MOVZBL  (NAMEPTR), R0                                   ; 7586
                                   6042 83  90 00068         MOVB    (CHARPTR)+, (R0)[NAMEPTR]                       ;
                                        55  D7 0006C         DECL    CHARLEN                                         ; 7588
                                        BB  11 0006E         BRB     3$                                              ; 7579
                                        62  95 00070 8$:     TSTB    (NAMEPTR)                                       ; 7595
                                        09  12 00072         BNEQ    9$                                              ;
                                        54  DD 00074         PUSHL   R4                                              ;
                 00000000G              00  FB 00076         CALLS   #1, DBG$SYNTAX_ERROR                            ;
                                    04  A4  53  D0 0007D 9$:  MOVL    CHARPTR, 4(R4)                                  ; 7596
                                        64  55  B0 00081         MOVW    CHARLEN, (R4)                                   ; 7597
                                        50  52  D0 00084         MOVL    NAMEPTR, R0                                     ; 7598
                                            04 00087         RET                                                     ; 7600

; Routine Size:  136 bytes,    Routine Base:  DBG$CODE + 3EAE
```

```
;  7525          7601  1 ROUTINE PARSE_WINDOW_PARAMETERS(INPUT_DESC, R_BEG, R_LEN, C_BEG, C_LEN): NOVALUE =
;  7526          7602  1 !
;  7527          7603  1 ! FUNCTION
;  7528          7604  1 !       This routine parses a "screen window specification" of the form
;  7529          7605  1 !       "(RBEG, RLEN, CBEG, CLEN)".  The parse pointer is assumed to point
;  7530          7606  1 !       to the first character after the opening left parenthesis when this
;  7531          7607  1 !       routine is called.  This routine picks up and returns the values of
;  7532          7608  1 !       all the window parameters, and leaves the parse pointer pointing to
;  7533          7609  1 !       the first character after the closing right parenthesis.  The CBEG
;  7534          7610  1 !       and CLEN parameters are optional in the input string, and are given
;  7535          7611  1 !       default values if omitted.
;  7536          7612  1 !
;  7537          7613  1 ! INPUTS
;  7538          7614  1 !       INPUT_DESC - The address of the input string descriptor for the command
;  7539          7615  1 !                    line to be parsed for a window specification.
;  7540          7616  1 !
;  7541          7617  1 !       R_BEG   - The address of a longword location to receive the RBEG value.
;  7542          7618  1 !
;  7543          7619  1 !       R_LEN   - The address of a longword location to receive the RLEN value.
;  7544          7620  1 !
;  7545          7621  1 !       C_BEG   - The address of a longword location to receive the CBEG value.
;  7546          7622  1 !
;  7547          7623  1 !       C_LEN   - The address of a longword location to receive the CLEN value.
;  7548          7624  1 !
;  7549          7625  1 ! OUTPUTS
;  7550          7626  1 !       INPUT_DESC - The INPUT_DESC string descriptor is updated to reflect the
;  7551          7627  1 !                    new parse location after picking up the window parameters.
;  7552          7628  1 !
;  7553          7629  1 !       R_BEG   - The value of the RBEG parameter (the beginning row location
;  7554          7630  1 !                 of the screen window) is returned to the R_BEG location.
;  7555          7631  1 !
;  7556          7632  1 !       R_LEN   - The value of the RLEN parameter (the row length or height
;  7557          7633  1 !                 of the screen window) is returned to the R_LEN location.
;  7558          7634  1 !
;  7559          7635  1 !       C_BEG   - The value of the CBEG parameter (the beginning column loca-
;  7560          7636  1 !                 tion of the screen window) is returned to the C_BEG location.
;  7561          7637  1 !
;  7562          7638  1 !       C_LEN   - The value of the CLEN parameter (the column length or width
;  7563          7639  1 !                 of the screen window) is returned to the C_LEN location.
;  7564          7640  1 !
;  7565          7641  1
;  7566          7642  2     BEGIN
;  7567          7643  2
;  7568          7644  2     MAP
;  7569          7645  2         R_BEG: REF VECTOR[1,LONG],      ! Return location for RBEG parameter
;  7570          7646  2         R_LEN: REF VECTOR[1,LONG],      ! Return location for RLEN parameter
;  7571          7647  2         C_BEG: REF VECTOR[1,LONG],      ! Return location for CBEG parameter
;  7572          7648  2         C_LEN: REF VECTOR[1,LONG];      ! Return location for CLEN parameter
;  7573          7649  2
;  7574          7650  2     LOCAL
;  7575          7651  2         CBEG,                           ! Beginning column location of window
;  7576          7652  2         CLEN,                           ! Column length (width) of window
;  7577          7653  2         RBEG,                           ! Beginning row location of window
;  7578          7654  2         RLEN;                           ! Row length (height) of window
;  7579          7655  2
;  7580          7656  2
;  7581          7657  2
```

```
;  7582          7658  2     ! Pick up the RBEG parameter (the beginning row location) and check it
;  7583          7659  2     ! for being within its valid range.
;  7584          7660  2     !
;  7585          7661  2     DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, RBEG);
;  7586          7662  3     IF (.RBEG LSS 1) OR (.RBEG GTR DBG$K_PASTE_SIZE - 2)
;  7587          7663  2     THEN
;  7588          7664  2         SIGNAL(DBG$_INVWINPAR, 1, .RBEG);
;  7589          7665  2
;  7590          7666  2
;  7591          7667  2     ! Pick up the RLEN parameter (the row length or height of the display) and
;  7592          7668  2     ! check it for being within its valid range.  A comma must precede it.
;  7593          7669  2     !
;  7594          7670  2     IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COMMA, 1)
;  7595          7671  2     THEN
;  7596          7672  2         DBG$SYNTAX_ERROR(.INPUT_DESC);
;  7597          7673  2
;  7598          7674  2     DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, RLEN);
;  7599          7675  3     IF (.RLEN LSS 1) OR (.RBEG + .RLEN GTR DBG$K_PASTE_SIZE - 1)
;  7600          7676  2     THEN
;  7601          7677  2         SIGNAL(DBG$_INVWINPAR, 1, .RLEN);
;  7602          7678  2
;  7603          7679  2
;  7604          7680  2     ! Next pick up the optional column location and width (CBEG and CLEN).
;  7605          7681  2     ! We do this only if there is not a closing parenthesis at this point.
;  7606          7682  2     !
;  7607          7683  2     CBEG = 1;
;  7608          7684  2     CLEN = 132;
;  7609          7685  2     IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_RPAREN, 1)
;  7610          7686  2     THEN
;  7611          7687  3         BEGIN
;  7612          7688  3         IF NOT .DBG$GL_DEVELOPER[7] THEN DBG$SYNTAX_ERROR(.INPUT_DESC); !<<-----
;  7613          7689  3
;  7614          7690  3
;  7615          7691  3         ! Pick up the CBEG parameter (the beginning column location) and check
;  7616          7692  3         ! it for being within its valid range.  A comma must precede this
;  7617          7693  3         ! parameter.
;  7618          7694  3         !
;  7619          7695  3         IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COMMA, 1)
;  7620          7696  3         THEN
;  7621          7697  3             DBG$SYNTAX_ERROR(.INPUT_DESC);
;  7622          7698  3
;  7623          7699  3         DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, CBEG);
;  7624          7700  4         IF (.CBEG LSS 1) OR (.CBEG GTR 132)
;  7625          7701  3         THEN
;  7626          7702  3             SIGNAL(DBG$_INVWINPAR, 1, .CBEG);
;  7627          7703  3
;  7628          7704  3
;  7629          7705  3         ! Pick up the CLEN parameter (the column length or width of the
;  7630          7706  3         ! display) and check it for being within its valid range.  A comma
;  7631          7707  3         ! must precede it.
;  7632          7708  3         !
;  7633          7709  3         IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_COMMA, 1)
;  7634          7710  3         THEN
;  7635          7711  3             DBG$SYNTAX_ERROR(.INPUT_DESC);
;  7636          7712  3
;  7637          7713  3         DBG$NSAVE_DECIMAL_INTEGER(.INPUT_DESC, CLEN);
;  7638          7714  4         IF (.CLEN LSS 1) OR (.CBEG + .CLEN GTR 133)
```

```
;  7639     7715  3          THEN
;  7640     7716  3              SIGNAL(DBG$_INVWINPAR, 1, .CLEN);
;  7641     7717  3
;  7642     7718  3
;  7643     7719  3          ! Finally make sure the closing parenthesis is there too.
;  7644     7720  3          !
;  7645     7721  3          IF NOT DBG$NMATCH(.INPUT_DESC, DBG$CS_RPAREN, 1)
;  7646     7722  3          THEN
;  7647     7723  3              DBG$SYNTAX_ERROR(.INPUT_DESC);
;  7648     7724  3
;  7649     7725  2          END;                                ! End of code for optional parameters
;  7650     7726  2
;  7651     7727  2
;  7652     7728  2      ! All window parameters have been parsed.  Return the parameter values we
;  7653     7729  2      ! found to the caller.  Then return.
;  7654     7730  2      !
;  7655     7731  2      R_BEG[0] = .RBEG;
;  7656     7732  2      R_LEN[0] = .RLEN;
;  7657     7733  2      C_BEG[0] = .CBEG;
;  7658     7734  2      C_LEN[0] = .CLEN;
;  7659     7735  2      RETURN;
;  7660     7736  2
;  7661     7737  1      END;


                                01FC 00000 PARSE_WINDOW_PARAMETERS:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8                ;  7601
58 00000000G   00  9E 00002                MOVAB   DBG$NSAVE_DECIMAL_INTEGER, R8
57 00000000G   00  9E 00009                MOVAB   LIB$SIGNAL, R7
56 00000000G   00  9E 00010                MOVAB   DBG$SYNTAX_ERROR, R6
55 00000000G   00  9E 00017                MOVAB   DBG$NMATCH, R5
54 00000000'   EF  9E 0001E                MOVAB   DBG$CS_COMMA, R4
5E             10  C2 00025                SUBL2   #16, SP
               5E  DD 00028                PUSHL   SP                                       ;  7661
52       04    AC  D0 0002A                MOVL    INPUT_DESC, R2
               52  DD 0002E                PUSHL   R2
68             02  FB 00030                CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER
53             6E  D0 00033                MOVL    RBEG, R3                                 ;  7662
               05  15 00036                BLEQ    1$
13             53  D1 00038                CMPL    R3, #19
               0D  15 0003B                BLEQ    2$
               53  DD 0003D 1$:            PUSHL   R3                                       ;  7664
               01  DD 0003F                PUSHL   #1
   00028AA2    8F  DD 00041                PUSHL   #166562
67             03  FB 00047                CALLS   #3, LIB$SIGNAL
               01  DD 0004A 2$:            PUSHL   #1                                       ;  7670
               54  BB 0004C                PUSHR   #^M<R2,R4>
65             53  FB 0004E                CALLS   #3, DBG$NMATCH
05             50  E8 00051                BLBS    R0, 3$
               52  DD 00054                PUSHL   R2                                       ;  7672
66             01  FB 00056                CALLS   #1, DBG$SYNTAX_ERROR
         04    AE  9F 00059 3$:            PUSHAB  RLEN                                     ;  7674
               52  DD 0005C                PUSHL   R2
68             02  FB 0005E                CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER
```

```
                        04   AE  D5 00061          TSTL    RLEN                                    ; 7675
                             0A  15 00064          BLEQ    4$
          50        53       04   AE  C1 00066          ADDL3   RLEN, R3, R0
                    14            50  D1 00068          CMPL    R0, #20
                                  0E  15 0006E          BLEQ    5$
                        04   AE  DD 00070 4$:       PUSHL   RLEN                                    ; 7677
                             01  DD 00073          PUSHL   #1
                00028AA2     8F  DD 00075          PUSHL   #166562
                    67            03  FB 0007B          CALLS   #3, LIB$S!GNAL
                08   AE            01  D0 0007E 5$:       MOVL    #1, CBEG                        ; 7683
                0C   AE       84   8F  9A 00082          MOVZBL  #132, CLEN                      ; 7684
                                   01  DD 00087          PUSHL   #1                              ; 7685
                        78   A4   9F  00089          PUSHAB  DBG$CS_RPAREN
                             52  DD 0008C          PUSHL   R2
                    65            03  FB 0008E          CALLS   #3, DBG$NMATCH
                    03            50  E9 00091          BLBC    R0, 6$
                            008C  31  00094          BRW     14$
                00000000G    00   95  00097 6$:       TSTB    DBG$GL_DEVELOPER                    ; 7688
                             05  19 0009D          BLSS    7$
                             52  DD 0009F          PUSHL   R2
                    66            01  FB 000A1          CALLS   #1, DBG$SYNTAX_ERROR
                             01  DD 000A4 7$:       PUSHL   #1                                  ; 7695
                             14  BB 000A6          PUSHR   #^M<R2,R4>
                    65            03  FB 000A8          CALLS   #3, DBG$NMATCH
                    05            50  E8 000AB          BLBS    R0, 8$
                             52  DD 000AE          PUSHL   R2                                  ; 7697
                    66            01  FB 000B0          CALLS   #1, DBG$SYNTAX_ERROR
                        08   AE   9F  000B3 8$:       PUSHAB  CBEG                                ; 7699
                             52  DD 000B6          PUSHL   R2
                    68            02  FB 000B8          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER
                        08   AE   D5  000BB          TSTL    CBEG                                ; 7700
                             0A  15 000BE          BLEQ    9$
          00000084  8F      08   AE  D1 000C0          CMPL    CBEG, #132
                             0E  15 000C8          BLEQ    10$
                        08   AE  DD 000CA 9$:       PUSHL   CBEG                                ; 7702
                             01  DD 000CD          PUSHL   #1
                00028AA2     8F  DD 000CF          PUSHL   #166562
                    67            03  FB 000D5          CALLS   #3, LIB$SIGNAL
                             01  DD 000D8 10$:      PUSHL   #1                                  ; 7709
                             14  BB 000DA          PUSHR   #^M<R2,R4>
                    65            03  FB 000DC          CALLS   #3, DBG$NMATCH
                    05            50  E8 000DF          BLBS    R0, 11$
                             52  DD 000E2          PUSHL   R2                                  ; 7711
                    66            01  FB 000E4          CALLS   #1, DBG$SYNTAX_ERROR
                        0C   AE   9F  000E7 11$:      PUSHAB  CLEN                                ; 7713
                             52  DD 000EA          PUSHL   R2
                    68            02  FB 000EC          CALLS   #2, DBG$NSAVE_DECIMAL_INTEGER
                        0C   AE   D5  000EF          TSTL    CLEN                                ; 7714
                             0F  15 000F2          BLEQ    12$
          50        08   AE       0C   AE  C1 000F4          ADDL3   CLEN, CBEG, R0
          00000085  8F            50  D1 000FA          CMPL    R0, #133
                                  0E  15 00101          BLEQ    13$
                        0C   AE  DD 00103 12$:      PUSHL   CLEN                                ; 7716
                             01  DD 00106          PUSHL   #1
                00028AA2     8F  DD 00108          PUSHL   #166562
                    67            03  FB 0010E          CALLS   #3, LIB$SIGNAL
                             01  DD 00111 13$:      PUSHL   #1                                  ; 7721
```

```
                        78    A4  9F 00113          PUSHAB  DBG$CS_RPAREN                    ;
                              52  DD 00116          PUSHL   R2                              ;
                    65        03  FB 00118          CALLS   #3, DBG$NMATCH                  ;
                    05        50  E8 0011B          BLBS    R0, 14$                         ;
                              52  DD 0011E          PUSHL   R2                              ; 7723
                    66        01  FB 00120          CALLS   #1, DBG$SYNTAX_ERROR            ;
                08  BC        53  D0 00123 14$:     MOVL    R3, @R_BEG                      ; 7731
                0C  BC    04  AE  D0 00127          MOVL    RLEN, @R_LEN                    ; 7732
                10  BC    08  AE  D0 0012C          MOVL    CBEG, @C_BEG                    ; 7733
                14  BC    0C  AE  D0 00131          MOVL    CLEN, @C_LEN                    ; 7734
                                  04 00136          RET                                     ; 7737

; Routine Size:  311 bytes,    Routine Base:  DBG$CODE + 3F36
```

```
;  7663     7738  1 ROUTINE REDIRECT_SCREEN_OUTPUT: NOVALUE =
;  7664     7739  1 !
;  7665     7740  1 ! FUNCTION
;  7666     7741  1 !       This routine redirects screen output to logical name DBG$OUTPUT if
;  7667     7742  1 !       a translation exists for this logical name.  Such redirection is
;  7668     7743  1 !       useful for directing DEBUG output to a second terminal, for example.
;  7669     7744  1 !       If no translation for DBG$OUTPUT exists, no redirection is done and
;  7670     7745  1 !       all screen output goes to SYS$OUTPUT.
;  7671     7746  1 !
;  7672     7747  1 ! INPUTS
;  7673     7748  1 !       NONE
;  7674     7749  1 !
;  7675     7750  1 ! OUTPUTS
;  7676     7751  1 !       NONE
;  7677     7752  1 !
;  7678     7753  1
;  7679     7754  2     BEGIN
;  7680     7755  2
;  7681     7756  2     LOCAL
;  7682     7757  2         LOGNAMDESC: BLOCK[8,BYTE],      ! String descriptor for DBG$OUTPUT name
;  7683     7758  2         RESULTBUF: VECTOR[80,BYTE],     ! Result string buffer for $TRNLOG
;  7684     7759  2         RESULTDESC: BLOCK[8,BYTE],      ! Result string descriptor for $TRNLOG
;  7685     7760  2         STATUS;                         ! Screen package status code
;  7686     7761  2
;  7687     7762  2
;  7688     7763  2
;  7689     7764  2     ! Set up the necessary descriptors and the attempt to translate logical
;  7690     7765  2     ! name DBG$OUTPUT.
;  7691     7766  2     !
;  7692     7767  2     LOGNAMDESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  7693     7768  2     LOGNAMDESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  7694     7769  2     LOGNAMDESC[DSC$W_LENGTH] = 10;
;  7695     7770  2     LOGNAMDESC[DSC$A_POINTER] = UPLIT BYTE(%ASCII 'DBG$OUTPUT');
;  7696     7771  2     RESULTDESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  7697     7772  2     RESULTDESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  7698     7773  2     RESULTDESC[DSC$W_LENGTH] = 80;
;  7699     7774  2     RESULTDESC[DSC$A_POINTER] = RESULTBUF;
;  7700     7775  2     STATUS = $TRNLOG(LOGNAM = LOGNAMDESC, RSLBUF = RESULTDESC);
;  7701     7776  2
;  7702     7777  2
;  7703     7778  2     ! If there is no translation, return immediately without redirecting the
;  7704     7779  2     ! screen output.
;  7705     7780  2     !
;  7706     7781  2     IF .STATUS EQL SS$_NOTRAN THEN RETURN;
;  7707     7782  2
;  7708     7783  2
;  7709     7784  2     ! There is a translation for DBG$OUTPUT.  Redirect screen output to
;  7710     7785  2     ! DBG$OUTPUT.  If the redirection fails for any reason, redirect output
;  7711     7786  2     ! back to SYS$OUTPUT.
;  7712     7787  2     !
;  7713     7788  2     STATUS = SCR$SET_OUTPUT(1, LOGNAMDESC);
;  7714     7789  2     IF NOT .STATUS
;  7715     7790  2     THEN
;  7716     7791  3         BEGIN
;  7717     7792  3         SIGNAL(DBG$_UNAOPESCR, 0, .STATUS);
;  7718     7793  3         LOGNAMDESC[DSC$A_POINTER] = UPLIT BYTE(%ASCII 'SYS$OUTPUT');
;  7719     7794  3         SCR$SET_OUTPUT(1, LOGNAMDESC);
```

```
;  7720          7795  2          END;
;  7721          7796  2
;  7722          7797  2
;  7723          7798  2      ! The redirection is completed.  Now return.
;  7724          7799  2      !
;  7725          7800  2      RETURN;
;  7726          7801  2
;  7727          7802  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                54  55  50  54  55  4F  24  47  42  44  00824 P.AIC:  .ASCII  \DBG$OUTPUT\                          ;
                54  55  50  54  55  4F  24  53  59  53  0082E P.AID:  .ASCII  \SYS$OUTPUT\                          ;

                                                                .EXTRN  SYS$TRNLOG

                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                    0004 00000 REDIRECT_SCREEN_OUTPUT:
                                                                .WORD   Save R2                                     ; 7738
                                    52 00000000G    00   9E 00002          MOVAB   SCR$SET_OUTPUT, R2             ;
                                    5E        A4    AE   9E 00009          MOVAB   -92(SP), SP                    ;
                                54  AE 010E000A     8F   D0 0000D          MOVL    #17694730, LOGNAMDESC          ; 7769
                                58  AE 00000000'    EF   9E 00015          MOVAB   P.AIC, LOGNAMDESC+4            ; 7770
                                    010E0050        8F   DD 0001D          PUSHL   #17694800                      ; 7773
                                04  AE        08    AE   9E 00023          MOVAB   RESULTBUF, RESULTDESC+4        ; 7774
                                                    7E   7C 00028          CLRQ    -(SP)                          ; 7775
                                                    7E   D4 0002A          CLRL    -(SP)                          ;
                                              0C    AE   9F 0002C          PUSHAB  RESULTDESC                     ;
                                                    7E   D4 0002F          CLRL    -(SP)                          ;
                                              6C    AE   9F 00031          PUSHAB  LOGNAMDESC                     ;
                          00000000G             00   06   FB 00034          CALLS   #6, SYS$TRNLOG                 ;
                          00000629              8F   50   D1 0003B          CMPL    STATUS, #1577                  ; 7781
                                                    2C   13 00042          BEQL    1$                             ;
                                              58    AE   9F 00044          PUSHAB  LOGNAMDESC                     ; 7788
                                                    01   DD 00047          PUSHL   #1                             ;
                                        62          02   FB 00049          CALLS   #2, SCR$SET_OUTPUT             ;
                                        21          50   E8 0004C          BLBS    STATUS, 1$                     ; 7789
                                                    50   DD 0004F          PUSHL   STATUS                         ; 7792
                                                    7E   D4 00051          CLRL    -(SP)                          ;
                                    00028FC3        8F   DD 00053          PUSHL   #167875                        ;
                          00000000G             00   03   FB 00059          CALLS   #3, LIB$SIGNAL                 ;
                                5C  AE 00000000'    EF   9E 00060          MOVAB   P.AID, LOGNAMDESC+4            ; 7793
                                              58    AE   9F 00068          PUSHAB  LOGNAMDESC                     ; 7794
                                                    01   DD 0006B          PUSHL   #1                             ;
                                        62          02   FB 0006D          CALLS   #2, SCR$SET_OUTPUT             ;
                                                    04 00070 1$:           RET                                    ; 7802

; Routine Size:  113 bytes,    Routine Base:  DBG$CODE + 406D
```

```
;  7729          7803  1 ROUTINE REGISTER_DISPLAY(DISPTR, REGVECTOR): NOVALUE =
;  7730          7804  1 !
;  7731          7805  1 ! FUNCTION
;  7732          7806  1 !       This routine generates the contents of a REGISTER screen display.
;  7733          7807  1 !       It accepts a pointer to a REGISTER Screen Display Entry and a
;  7734          7808  1 !       pointer to a vector of register values as input  and generates
;  7735          7809  1 !       the full contents of that screen display using the register values
;  7736          7810  1 !       in the input register value vector.
;  7737          7811  1 !
;  7738          7812  1 ! INPUTS
;  7739          7813  1 !       DISPTR  - A pointer to the Screen Display Entry for the register
;  7740          7814  1 !                 screen display to be generated.
;  7741          7815  1 !
;  7742          7816  1 !       REGVECTOR - A pointer to a register vector containing the register
;  7743          7817  1 !                 values to be printed in the DISPTR register screen display.
;  7744          7818  1 !                 The register vector is a vector of 17 longwords containing
;  7745          7819  1 !                 the values of registers R0 - R11, AP, FP, SP, PC, and PSL
;  7746          7820  1 !                 in that order.
;  7747          7821  1 !
;  7748          7822  1 ! OUTPUTS
;  7749          7823  1 !       NONE
;  7750          7824  1 !
;  7751          7825  1
;  7752          7826  2     BEGIN
;  7753          7827  2
;  7754          7828  2     MAP
;  7755          7829  2         DISPTR: REF DBG$DISP_ENTRY,     ! Pointer to Screen Display Entry
;  7756          7830  2         REGVECTOR: REF VECTOR[,LONG];   ! Pointer to register value vector
;  7757          7831  2
;  7758          7832  2     BUILTIN
;  7759          7833  2         PROBER;                         ! Probe a location for read access
;  7760          7834  2
;  7761          7835  2     LOCAL
;  7762          7836  2         APVAL: REF VECTOR[,LONG],       ! The Argument Pointer value (AP)
;  7763          7837  2         BLINK: REF DBG$DLINE_ENTRY,     ! Pointer to previous Display Line Entry
;  7764          7838  2         DLEPTR: REF DBG$DLINE_ENTRY,    ! Pointer to current Display Line Entry
;  7765          7839  2         FLINK: REF DBG$DLINE_ENTRY,     ! Pointer to next Display Line Entry
;  7766          7840  2         HEXDESC: BLOCK[8,BYTE],         ! String descriptor for HEXSTRING
;  7767          7841  2         HEXSTRING: VECTOR[8,BYTE],      ! Hexadecimal format of register value
;  7768          7842  2         LINE_TBL: VECTOR[5, LONG],      ! Table of pointer to Display Lines
;  7769          7843  2         MSG_LENGTH,                     ! Length of returned message text
;  7770          7844  2         MSG_DESCR: BLOCK[8,BYTE],       ! Message descriptor for $GETMSG
;  7771          7845  2         PSLVAL,                         ! The Processor Status Longword value
;  7772          7846  2         REND,                           ! Rendition code for changed values
;  7773          7847  2         SPVAL: REF VECTOR[,LONG],       ! The Stack Pointer value (SP)
;  7774          7848  2         STATUS,                         ! Status returned by $GETMSG
;  7775          7849  2         TPTR: REF VECTOR[,BYTE];        ! Pointer to text contents for a line
;  7776          7850  2
;  7777          7851  2
;  7778          7852  2
;  7779          7853  2     ! Check the DISPTR input parameter for validity.
;  7780          7854  2     !
;  7781          7855  2     IF .DISPTR[DBG$B_DISP_KIND] NEQ DBG$K_DISP_REGISTER
;  7782          7856  2     THEN
;  7783          7857  2         $DBG_ERROR('DBGSCREEN\REGISTER_DISPLAY 10');
;  7784          7858  2
;  7785          7859  2     IF (.DISPTR[DBG$W_DISP_LINECNT] NEQ 0) AND
```

```
;  7786          7860  3        (.DISPTR[DBG$W_DISP_LINECNT] NEQ 5)
;  7787          7861  2    THEN
;  7788          7862  2        $DBG_ERROR('DBGSCREEN\REGISTER_DISPLAY 20');
;  7789          7863  2
;  7790          7864  2
;  7791          7865  2    ! If this register is empty (meaning that it does not yet contain any text
;  7792          7866  2    ! lines), fill in the register display template.  If text lines exist, they
;  7793          7867  2    ! are assumed to contain the original template with register values filled
;  7794          7868  2    ! in, so that the template does not need to be regenerated.
;  7795          7869  2    !
;  7796          7870  2    REND = .DISPTR[DBG$B_DISP_REND] XOR
;  7797          7871  2                        (DBG$M_DISP_REND_RV OR DBG$M_DISP_REND_BLD);
;  7798          7872  2    IF .DISPTR[DBG$W_DISP_LINECNT] EQL 0
;  7799          7873  2    THEN
;  7800          7874  3        BEGIN
;  7801          7875  3        REND = .DISPTR[DBG$B_DISP_REND];
;  7802          7876  3        DISPTR[DBG$W_DISP_DROW] = 1;
;  7803          7877  3
;  7804          7878  3
;  7805          7879  3        ! Loop to generate the templates for the five lines of the display.
;  7806          7880  3        !
;  7807          7881  3        INCR I FROM 0 TO 4 DO
;  7808          7882  4            BEGIN
;  7809          7883  4
;  7810          7884  4
;  7811          7885  4            ! Allocate the Display Line Entry for the current line and set up
;  7812          7886  4            ! its contents including the text of the line.
;  7813          7887  4            !
;  7814          7888  4            DLEPTR = DBG$GET_MEMORY(DBG$K_DLINE_ENTSIZE + (80*2)/%UPVAL + 1);
;  7815          7889  4            DLEPTR[DBG$B_DLINE_LENGTH] = 80;
;  7816          7890  4            DLEPTR[DBG$B_DLINE_REND] = .DISPTR[DBG$B_DISP_REND];
;  7817          7891  4            CASE .I FROM 0 TO 4 OF
;  7818          7892  4                SET
;  7819          7893  4
;  7820          7894  4                [0]:
;  7821          7895  4                    TPTR = UPLIT BYTE(80, %ASCII 'R0: xxxxxxxx R1: xxxxxxxx ',
;  7822          7896  4                     'R2: xxxxxxxx R3: xxxxxxxx  @AP:xxxxxxxx  @SP:xxxxxxxx ');
;  7823          7897  4
;  7824          7898  4                [1]:
;  7825          7899  4                    TPTR = UPLIT BYTE(80, %ASCII 'R4: xxxxxxxx R5: xxxxxxxx ',
;  7826          7900  4                     'R6: xxxxxxxx R7: xxxxxxxx   +4:xxxxxxxx   +4:xxxxxxxx ');
;  7827          7901  4
;  7828          7902  4                [2]:
;  7829          7903  4                    TPTR = UPLIT BYTE(80, %ASCII 'R8: xxxxxxxx R9: xxxxxxxx ',
;  7830          7904  4                     'R10:xxxxxxxx R11:xxxxxxxx   +8:xxxxxxxx   +8:xxxxxxxx ');
;  7831          7905  4
;  7832          7906  4                [3]:
;  7833          7907  4                    TPTR = UPLIT BYTE(80, %ASCII 'AP: xxxxxxxx FP: xxxxxxxx ',
;  7834          7908  4                     'SP: xxxxxxxx PC: xxxxxxxx  +12:xxxxxxxx  +12:xxxxxxxx ');
;  7835          7909  4
;  7836          7910  4                [4]:
;  7837          7911  4                    TPTR = UPLIT BYTE(80, %ASCII '                                  ',
;  7838          7912  4                     '                                          N:x Z:x V:x C:x ');
;  7839          7913  4
;  7840          7914  4                TES;
;  7841          7915  4
;  7842          7916  4            CH$MOVE(81, .TPTR, DLEPTR[DBG$A_DLINE_TEXT]);
```

```
;  7843          7917  4
;  7844          7918  4
;  7845          7919  4              ! Link the new Display Line Entry into the line entry chain for
;  7846          7920  4              ! the DISPTR display.  Also increment the display's line count.
;  7847          7921  4              ! Then loop for the next line.
;  7848          7922  4              !
;  7849          7923  4              FLINK = DISPTR[DBG$L_DISP_START_LINE_PTR];
;  7850          7924  4              BLINK = .FLINK[DBG$L_DLINE_BLINK];
;  7851          7925  4              DLEPTR[DBG$L_DLINE_FLINK] = .FLINK;
;  7852          7926  4              DLEPTR[DBG$L_DLINE_BLINK] = .BLINK;
;  7853          7927  4              FLINK[DBG$L_DLINE_BLINK] = .DLEPTR;
;  7854          7928  4              BLINK[DBG$L_DLINE_FLINK] = .DLEPTR;
;  785'          7929  4              DISPTR[DBG$W_DISP_LINECNT] = .DISPTR[DBG$W_DISP_LINECNT] + 1;
;  785(          7930  4              IF .I EQL 0 THEN DISPTR[DBG$L_DISP_WINDOW_PTR] = .DLEPTR;
;  7857          7931  4
;  7858          7932  3              END;                            ! End of line initialization loop
;  7859          7933  3
;  7860          7934  2          END;                                ! End of display template initialization
;  7861          7935  2
;  7862          7936  2
;  7863          7937  2      ! Set up the line table with pointer to the five display lines.  Also clear
;  7864          7938  2      ! out the line's rendition vector (which immediately follows the text).
;  786)          7939  2      !
;  7866          7940  2      DLEPTR = .DISPTR[DBG$L_DISP_START_LINE_PTR];
;  7867          7941  2      INCR I FROM 0 TO 4 DO
;  7868          7942  3          BEGIN
;  7869          7943  3          LINE_TBL[.I] = .DLEPTR;
;  7870          7944  3          TPTR = DLEPTR[DBG$A_DLINE_TEXT];
;  7871          7945  3          CH$FILL(.DISPTR[DBG$B_DISP_REND], 80, TPTR[81]);
;  7872          7946  3          DLEPTR[DBG$V_DLINE_RENDFLG] = FALSE;
;  7873          7947  3          DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  7874          7948  2          END;
;  7875          7949  2
;  7876          795v  .
;  7877          7951  2      ! Set up the string descriptor used for hexadecimal formatting.
;  7878          7952  2      !
;  7879          7953  2      HEXDESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  7880          7954  2      HEXDESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  7881          7955  2      HEXDESC[DSC$W_LENGTH] = 8;
;  7882          7956  2      HEXDESC[DSC$A_POINTER] = HEXSTRING;
;  7883          7957  2
;  7884          7958  2
;  7885          7959  2      ! Fill all the register values into their respective slots in the register
;  7886          7960  2      ! display.  All values are formatted in hexadecimal.
;  7887          7961  2      !
;  7888          7962  2      INCR I FROM 0 TO 15 DO
;  7889          7963  3          BEGIN
;  7890          7964  3          OTS$CVT_L_TZ(REGVECTOR[.I], HEXDESC, 8);
;  7891          7965  3          DLEPTR = .LINE_TBL[.I/4];
;  7892          7966  3          TPTR = DLEPTR[DBG$A_DLINE_TEXT];
;  7893          7967  3          REGISTER_FILL(.DLEPTR, 8, HEXSTRING, TPTR[(.I MOD 4)*13 + 5], .REND);
;  7894          7968  2          END;
;  7895          7969  2
;  7896          7970  2
;  7897          7971  2      ! Fill the first few values from the Argument Vector and from the top of
;  7898          7972  2      ! the stack into the register display.
;  7899          7973  2      !
```

```
; 7900          7974  2     APVAL = .REGVECTOR[12];
; 7901          7975  2     SPVAL = .REGVECTOR[14];
; 7902          7976  2     INCR I FROM 0 TO 3 DO
; 7903          7977  3         BEGIN
; 7904          7978  3         DLEPTR = .LINE_TBL[.I];
; 7905          7979  3         TPTR = DLEPTR[DBG$A_DLINE_TEXT];
; 7906          7980  3
; 7907          7981  3
; 7908          7982  3         ! Fill the I-th value of the Argument Vector into the register display.
; 7909          7983  3         ! If that element is not readable, use "-noread-" instead.
; 7910          7984  3         !
; 7911          7985  3         IF PROBER(%REF(0), %REF(4), APVAL[.I])
; 7912          7986  3         THEN
; 7913          7987  3             OTS$CVT_L_TZ(APVAL[.I], HEXDESC, 8)
; 7914          7988  3
; 7915          7989  3         ELSE
; 7916          7990  3             CH$MOVE(8, UPLIT BYTE('-noread-'), HEXSTRING);
; 7917          7991  3
; 7918          7992  3         REGISTER_FILL(.DLEPTR, 8, HEXSTRING, TPTR[58], .REND);
; 7919          7993  3
; 7920          7994  3
; 7921          7995  3         ! Fill the I-th value off the Stack Pointer (SP) into the register
; 7922          7996  3         ! display.  If that element is not readable, use "-noread-" instead.
; 7923          7997  3         !
; 7924          7998  3         IF PROBER(%REF(0), %REF(4), SPVAL[.I])
; 7925          7999  3         THEN
; 7926          8000  3             OTS$CVT_L_TZ(SPVAL[.I], HEXDESC, 8)
; 7927          8001  3
; 7928          8002  3         ELSE
; 7929          8003  3             CH$MOVE(8, UPLIT BYTE('-noread-'), HEXSTRING);
; 7930          8004  3
; 7931          8005  3         REGISTER_FILL(.DLEPTR, 8, HEXSTRING, TPTR[72], .REND);
; 7932          8006  3
; 7933          8007  2         END;                                ! End of loop for @AP+x and @SP+x
; 7934          8008  2
; 7935          8009  2
; 7936          8010  2     ! Fill in the message text (if any) that corresponds to the value in R0.
; 7937          8011  2     ! The message text is truncated to the first 60 characters.
; 7938          8012  2     !
; 7939          8013  2     DLEPTR = .LINE_TBL[4];
; 7940          8014  2     TPTR = DLEPTR[DBG$A_DLINE_TEXT];
; 7941          8015  2     CH$FILL(' ', 60, TPTR[1]);
; 7942          8016  2     MSG_DESCR[DSC$B_CLASS] = DSC$K_CLASS_S;
; 7943          8017  2     MSG_DESCR[DSC$B_DTYPE] = DSC$K_DTYPE_T;
; 7944          8018  2     MSG_DESCR[DSC$W_LENGTH] = 60;
; 7945          8019  2     MSG_DESCR[DSC$A_POINTER] = TPTR[1];
; 7946        P 8020  2     STATUS = $GETMSG(MSGID = .REGVECTOR[0],
; 7947          8021  2                      MSGLEN = MSG_LENGTH, BUFADR = MSG_DESCR, FLAGS = 0);
; 7948          8022  3     IF (.STATUS EQL SS$_MSGNOTFND) OR (NOT .STATUS)
; 7949          8023  2     THEN
; 7950          8024  2         CH$FILL(' ', 60, TPTR[1]);
; 7951          8025  2
; 7952          8026  2
; 7953          8027  2     ! Fill in the values of the four condition codes (N, Z, V, and C) into the
; 7954          8028  2     ! last line of the register display.
; 7955          8029  2     !
; 7956          8030  2     PSLVAL = .REGVECTOR[16];
```

```
;  7957          8031  2     HEXSTRING[0] = .PSLVAL<V'(3)> + '0';
;  7958          8032  2     REGISTER_FILL(.DLEPTR, 1, HEXSTRING, TPTR[67], .REND);
;  7959          8033  2     HEXSTRING[0] = .PSLVAL<V'(2)> + '0';
;  7960          8034  2     REGISTER_FILL(.DLEPTR, 1, HEXSTRING, TPTR[71], .REND);
;  7961          8035  2     HEXSTRING[0] = .PSLVAL<V'(1)> + '0';
;  7962          8036  2     REGISTER_FILL(.DLEPTR, 1, HEXSTRING, TPTR[75], .REND);
;  7963          8037  2     HEXSTRING[0] = .PSLVAL<V'(0)> + '0';
;  7964          8038  2     REGISTER_FILL(.DLEPTR, 1, HEXSTRING, TPTR[79], .REND);
;  7965          8039  2
;  7966          8040  2
;  7967          8041  2     ! The register display is fully generated.  Now return.
;  7968          8042  2     !
;  7969          8043  2     RETURN;
;  7970          8044  2
;  7971          8045  1     END;


                                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

49 47 45 52 5C 4E 45 45 52 43 53 47 42 44 1D  00838 P.AIE:  .ASCII  <29>\DBGSCREEN\<92>\REGISTER_DISPLAY 1\      ;
   31 20 59 41 4C 50 53 49 44 5F 52 45 54 53  00847                                                            ;
                                              00855          .ASCII  \0\                                       ;
49 47 45 52 5C 4E 45 45 52 43 53 47 42 44 1D  00856 P.AIF:  .ASCII  <29>\DBGSCREEN\<92>\REGISTER_DISPLAY 2\      ;
   32 20 59 41 4C 50 53 49 44 5F 52 45 54 53  00865                                                            ;
                                              00873          .ASCII  \0\                                       ;
                                              00874 P.AIG:   .BYTE   80                                        ;
31 52 20 78 78 78 78 78 78 78 78 20 3A 30  52  00875          .ASCII  \R0: xxxxxxxx R1: xxxxxxxx \              ;
            20 78 78 78 78 78 78 78 78 20  3A  00884                                                            ;
33 52 20 78 78 78 78 78 78 78 78 20 3A 32  52  0088F          .ASCII  \R2: xxxxxxxx R3: xxxxxxxx  @AP:xxxxxxxx \ ;
50 41 40 20 20 78 78 78 78 78 78 78 78 20  3A  0089E                                                            ;
            20 78 78 78 78 78 78 78 78 3A  3A  008AD                                                            ;
   20 78 78 78 78 78 78 78 78 3A 50 53 40  20  008B7          .ASCII  \ @SP:xxxxxxxx \                          ;
                                              008C5 P.AIH:   .BYTE   80                                        ;
35 52 20 78 78 78 78 78 78 78 78 20 3A 34  52  008C6          .ASCII  \R4: xxxxxxxx R5: xxxxxxxx \              ;
            20 78 78 78 78 78 78 78 78 20  3A  008D5                                                            ;
37 52 20 78 78 78 78 78 78 78 78 20 3A 36  52  008E0          .ASCII  \R6: xxxxxxxx R7: xxxxxxxx   +4:xxxxxxxx \ ;
34 2B 20 20 20 78 78 78 78 78 78 78 78 20  3A  008EF                                                            ;
            20 78 78 78 78 78 78 78 78 3A  3A  008FE                                                            ;
   20 78 78 78 78 78 78 78 78 3A 34 2B 20  20  00908          .ASCII  \  +4:xxxxxxxx \                          ;
                                              00916 P.AII:   .BYTE   80                                        ;
39 52 20 78 78 78 78 78 78 78 78 20 3A 38  52  00917          .ASCII  \R8: xxxxxxxx R9: xxxxxxxx \              ;
            20 78 78 78 78 78 78 78 78 20  3A  00926                                                            ;
31 52 20 78 78 78 78 78 78 78 78 31 30 31  52  00931          .ASCII  \R10:xxxxxxxx R11:xxxxxxxx   +8:xxxxxxxx \ ;
38 2B 20 20 20 78 78 78 78 78 78 78 78 3A  31  00940                                                            ;
            20 78 78 78 78 78 78 78 78 3A  3A  0094F                                                            ;
   20 78 78 78 78 78 78 78 78 3A 38 2B 20  20  00959          .ASCII  \  +8:xxxxxxxx \                          ;
                                              00967 P.AIJ:   .BYTE   80                                        ;
50 46 20 78 78 78 78 78 78 78 78 20 3A 50  41  00968          .ASCII  \AP: xxxxxxxx FP: xxxxxxxx \              ;
            20 78 78 78 78 78 78 78 78 20  3A  00977                                                            ;
43 50 20 78 78 78 78 78 78 78 78 20 3A 50  53  00982          .ASCII  \SP: xxxxxxxx PC: xxxxxxxx  +12:xxxxxxxx \ ;
32 31 2B 20 20 78 78 78 78 78 78 78 78 20  3A  00991                                                            ;
            20 78 78 78 78 78 78 78 78 3A  3A  009A0                                                            ;
   20 78 78 78 78 78 78 78 78 3A 32 31 2B  20  009AA          .ASCII  \ +12:xxxxxxxx \                          ;
                                              009B8 P.AIK:   .BYTE   80                                        ;
20 20 20 20 20 20 20 20 20 20 20 20 20 20  20  009B9          .ASCII  \                          \              ;
               20 20 20 20 20 20 20 20 20 20  00C8                                                           ;
```

```
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 009D3          .ASCII  \                                   N:\ ;
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 009E2                                                        ;
                  3A 4E 20 20 20 20 20 20 20 20 009F1                                                     ;
   20 78 3A 43 20 78 3A 56 20 78 3A 5A 20 78 009FB          .ASCII  \x Z:x V:x C:x \                       ;
                  2D 64 61 65 72 6F 6E 2D 00A09 P.AIL:     .ASCII  \-noread-\                             ;
                  2D 64 61 65 72 6F 6E 2D 00A11 P.AIM:     .ASCII  \-noread-\                             ;

                                                           .EXTRN  SYS$GETMSG

                                                           .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                        0FFC 00000 REGISTER_DISPLAY:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11   ; 7803
                                5E         38  C2 00002     SUBL2   #56, SP                                ;
                                56    04   AC  D0 00005     MOVL    DISPTR, R6                             ; 7855
                                04    08   A6  91 00009     CMPB    8(R6), #4                              ;
                                           15  13 0000D     BEQL    1$                                     ;
                                00000000'  EF  9F 0000F     PUSHAB  P.AIE                                  ; 7857
                                           01  DD 00015     PUSHL   #1                                     ;
                                00028362   8F  DD 00017     PUSHL   #164706                                ;
                        000( ..0G  00      03  FB 0001D     CALLS   #3, LIB$SIGNAL                         ;
                                52    1E   A6  3C 00024 1$: MOVZWL  30(R6), R2                            ; 7859
                                           1A  13 00028     BEQL    2$                                     ;
                                05         52  B1 0002A     CMPW    R2, #5                                 ; 7860
                                           15  13 0002D     BEQL    2$                                     ;
                                00000000'  EF  9F 0002F     PUSHAB  P.AIF                                  ; 7862
                                           01  DD 00035     PUSHL   #1                                     ;
                                00028362   8F  DD 00037     PUSHL   #164706                                ;
                           00000000G  00   03  FB 0003D     CALLS   #3, LIB$SIGNAL                         ;
                                50    09   A6  9A 00044 2$: MOVZBL  9(R6), R0                             ; 7871
                             6E    50   03  CD 00048     XORL3   #3, R0, REND                           ;
                                52   D5 0004C     TSTL    R2                                             ; 7872
                                03   13 0004E     BEQL    3$                                             ;
                                008A 31 00050     BRW     13$                                            ;
                                6E    09   A6  9A 00053 3$: MOVZBL  9(R6), REND                           ; 7875
                          18    A6        01  B0 00057     MOVW    #1, 24(R6)                             ; 7876
                          04    AE   20   A6  9E 0005B     MOVAB   32(R6), 4(SP)                          ; 7923
                                5A   D4 00060     CLRL    I                                              ;
                                2C   DD 00062 4$: PUSHL   #44                                            ; 7888
                           00000000G  00   01  FB 00064     CALLS   #1, DBG$GET_MEMORY                     ;
                                59   50 D0 0006B     MOVL    R0, DLEPTR                                     ;
                          0A    A9   50   8F  90 0006E     MOVB    #80, 10(DLEPTR)                        ; 7889
                          08    A9   09   A6  90 00073     MOVB    9(R6), 8(DLEPTR)                       ; 7890
                       04          00   5A  CF 00078     CASEL   I, #0, #4                              ; 7891
                0025        001C        0013      000A    0007C 5$: .WORD   6$-5$,-                        ;
                                                  002E    00084           7$-5$,-                        ;
                                                                          8$-5$,-                        ;
                                                                          9$-5$,-                        ;
                                                                          10$-5$                         ;
                                58 00000000'  EF  9E 00086 6$: MOVAB   P.AIG, TPTR                       ; 7895
                                              22  11 0008D     BRB     11$                                    ;
                                58 00000000'  EF  9E 0008F 7$: MOVAB   P.AIH, TPTR                       ; 7899
                                              19  11 00096     BRB     11$                                    ;
                                58 00000000'  EF  9E 00098 8$: MOVAB   P.AII, TPTR                       ; 7903
                                              10  11 0009F     BRB     11$                                    ;
                                58 00000000'  EF  9E 000A1 9$: MOVAB   P.AIJ, TPTR                       ; 7907
                                              07  11 000A8     BRB     11$                                    ;
```

```
                                58 00000000'   EF  9E 000AA 10$:    MOVAB   P.AIK, TPTR                             ; 7911
                             0C    A9          68      0051  11$:    MOVC3   #81, (TPTR), 12(DLEPTR)                 ; 7916
                                57       04    AE  D0 000B8          MOVL    4(SP), FLINK                            ; 7923
                                5B       04    A7  D0 000BC          MOVL    4(FLINK), BLINK                         ; 7924
                                69              57  D0 000C0          MOVL    FLINK, (DLEPTR)                         ; 7925
                             04    A9          5B  D0 000C3          MOVL    BLINK, 4(DLEPTR)                        ; 7926
                             04    A7          59  D0 000C7          MOVL    DLEPTR, 4(FLINK)                        ; 7927
                                6B              59  D0 000CB          MOVL    DLEPTR, (BLINK)                         ; 7928
                                      1E    A6  B6 000CE          INCW    30(R6)                                  ; 7929
                                            5A  D5 000D1          TSTL    I                                       ; 7930
                                            04  12 000D3          BNEQ    12$                                     ;
                             28    A6          59  D0 000D5          MOVL    DLEPTR, 40(R6)                          ;
                          85                5A  04  F3 000D9 12$:    AOBLEQ  #4, I, 4$                       ; 7881
                                59       20    A6  D0 000DD 13$:    MOVL    32(R6), DLEPTR                          ; 7940
                                            57  D4 000E1          CLRL    I                                       ; 7941
                          14 AE47           59  D0 000E3 14$:    MOVL    DLEPTR, LINE_TBL[I]                     ; 7943
                                58       0C    A9  9E 000E8          MOVAB   12(R9), TPTR                            ; 7944
                                50       09    A6  9A 000EC          MOVZBL  9(R6), R0                               ; 7945
    0050    8F                  50          6E      00  2C 000F0          MOVC5   #0, (SP), R0, #80, 81(TPTR)             ;
                                      51    A8      000F7                                                           ;
                             09    A9          01  8A 000F9          BICB2   #1, 9(DLEPTR)                           ; 7946
                                59              69  D0 000FD          MOVL    (DLEPTR), DLEPTR                        ; 7947
                          DF                57  04  F3 00100          AOBLEQ  #4, I, 14$                      ; 7941
                             30    AE 010E0008   8F  D0 00104          MOVL    #17694728, HEXDESC                      ; 7955
                             34    AE       28   AE  9E 0010C          MOVAB   HEXSTRING, HEXDESC+4                    ; 7956
                                56       08    AC  D0 00111          MOVL    REGVECTOR, R6                           ; 7964
                                            52  D4 00115          CLRL    I                                       ; 7967
                                            08  DD 00117 15$:    PUSHL   #8                                      ; 7964
                                      34    AE  9F 00119          PUSHAB  HEXDESC                                 ;
                                   6642     DF  0011C          PUSHAL  (R6)[I]                                 ;
                          00000000G         00  03  FB 0011F          CALLS   #3, OTS$CVT_L_TZ                        ;
                          50                52  04  C7 00126          DIVL3   #4, I, R0                       ; 7965
                                59       14 AE40  D0 0012A          MOVL    LINE_TBL[R0], DLEPTR                    ;
                                58       0C    A9  9E 0012F          MOVAB   12(R9), TPTR                            ; 7966
                                            6E  DD 00133          PUSHL   REND                                    ; 7967
    7E                          00          52      01  7A 00135          EMUL    #1, I, #0, -(SP)                        ;
    50                          50          8E      04  7B 0013A          EDIV    #4, (SP)+, R0, R0                       ;
                                50          0D      C4  0013F          MULL2   #13, R0                                 ;
                                      05 A048   9F  00142          PUSHAB  5(R0)[TPTR]                             ;
                                      30    AE  9F  00146          PUSHAB  HEXSTRING                               ;
                                            08  DD 00149          PUSHL   #8                                      ;
                                            59  DD 0014B          PUSHL   DLEPTR                                  ;
                                0000V       CF      05  FB 0014D          CALLS   #5, REGISTER_FILL                       ;
                          C1                52      0F  F3 00152          AOBLEQ  #15, I, 15$                     ; 7962
                                5B       30    A6  D0 00156          MOVL    48(R6), APVAL                           ; 7974
                                5A       38    A6  D0 0015A          MOVL    56(R6), SPVAL                           ; 7975
                                            57  D4 0015E          CLRL    I                                       ; 7998
                                59       14 AE47  D0 00160 16$:    MOVL    LINE_TBL[I], DLEPTR                     ; 7978
                                58       0C    A9  9E 00165          MOVAB   12(R9), TPTR                            ; 7979
                                      6B47      DF  00169          PUSHAL  (APVAL)[I]                              ; 7985
                          9E                04      00  0C 0016C          PROBER  #0, #4, @(SP)+                          ;
                                            11  13 00170          BEQL    17$                                     ;
                                            08  DD 00172          PUSHL   #8                                      ; 7987
                                      34    AE  9F  00174          PUSHAB  HEXDESC                                 ;
                                      6B47      DF  00177          PUSHAL  (APVAL)[I]                              ;
                             00000000G          00  03  FB 0017A          CALLS   #3, OTS$CVT_L_TZ                        ;
                                            09  11 00181          BRB     18$                                     ;
```

```
                28  AE 00C00000'  EF                08  28 00183 17$:   MOVC3   #8, P.AIL, HEXSTRING                    ; 7990
                                                    6E  DD 0018C 18$:   PUSHL   REND                                    ; 7992
                                                3A  A8  9F 0018E        PUSHAB  58(TPTR)                                ;
                                                30  AE  9F 00191        PUSHAB  HEXSTRING                               ;
                                                    08  DD 00194        PUSHL   #8                                      ;
                                                    59  DD 00196        PUSHL   DLEPTR                                  ;
                          0000V  CF                 05  FB 00198        CALLS   #5, REGISTER_FILL                       ;
                                                  6A47  DF 0019D        PUSHAL  (SPVAL)[I]                              ; 799F
                     9E           04                00  0C 001A0        PROBER  #0, #4, @(SP)+                          ;
                                                    11  13 001A4        BEQL    19$                                     ;
                                                    08  DD 001A6        PUSHL   #8                                      ; 8000
                                                34  AE  9F 001A8        PUSHAB  HEXDESC                                 ;
                                                  6A47  DF 001AB        PUSHAL  (SPVAL)[I]                              ;
                   00000000G  00                    03  FB 001AE        CALLS   #3, OTS$CVT_L_TZ                        ;
                                                    09  11 001B5        BRB     20$                                     ;
                28  AE 00000000'  EF                08  28 001B7 19$:   MOVC3   #8, P.AIM, HEXSTRING                   ; 8003
                                                    6E  DD 001C0 20$:   PUSHL   REND                                    ; 8005
                                                48  A8  9F 001C2        PUSHAB  72(TPTR)                                ;
                                                30  AE  9F 001C5        PUSHAB  HEXSTRING                               ;
                                                    08  DD 001C8        PUSHL   #8                                      ;
                                                    59  DD 001CA        PUSHL   DLEPTR                                  ;
                          0000V  CF                 05  FB 001CC        CALLS   #5, REGISTER_FILL                       ;
                     8B           57                03  F3 001D1        AOBLEQ  #3, I, 16$                              ; 7976
                                  59           24   AE  D0 001D5        MOVL    LINE_TBL+16, DLEPTR                     ; 8013
                                  58           0C   A9  9E 001D9        MOVAB   12(R9), TPTR                            ; 8014
          3C         20           6E                00  2C 001DD        MOVC5   #0, (SP), #32, #60, 1(TPTR)             ; 8015
                                                01  A8      001E2
                        0C  AE 010E003C         8F  D0 001E4        MOVL    #17694780, MSG_DESCR                    ; 8018
                        10  AE           01     A8  9E 001EC        MOVAB   1(TPTR), MSG_DESCR+4                    ; 8019
                                                    7E  7C 001F1        CLRQ    -(SP)                                   ; 8021
                                                14  AE  9F 001F3        PUSHAB  MSG_DESCR                               ;
                                                14  AE  9F 001F6        PUSHAB  MSG_LENGTH                              ;
                                                    66  DD 001F9        PUSHL   (R6)                                    ;
                   00000000G  00                    05  FB 001FB        CALLS   #5, SYS$GETMSG                          ;
                   00000621   8F                    50  D1 00202        CMPL    STATUS, #1569                           ; 8022
                                                    03  13 00209        BEQL    21$                                     ;
                                  07                50  E8 0020B        BLBS    STATUS, 22$                             ;
          3C         20           6E                00  2C 0020E 21$:   MOVC5   #0, (SP), #32, #60, 1(TPTR)           ; 8024
                                                01  A8      00213
                                  52           40   A6  D0 00215 22$:   MOVL    64(R6), PSLVAL                          ; 8030
          50         52           01                03  EF 00219        EXTZV   #3, #1, PSLVAL, R0                      ; 8031
                28   AE           50                30  81 0021E        ADDB3   #48, R0, HEXSTRING                      ;
                                                    6E  DD 00223        PUSHL   REND                                    ; 8032
                                                43  A8  9F 00225        PUSHAB  67(TPTR)                                ;
                                                30  AE  9F 00228        PUSHAB  HEXSTRING                               ;
                                                    01  DD 0022B        PUSHL   #1                                      ;
                                                    59  DD 0022D        PUSHL   DLEPTR                                  ;
                          0000V  CF                 05  FB 0022F        CALLS   #5, REGISTER_FILL                       ;
          50         52           01                02  EF 00234        EXTZV   #2, #1, PSLVAL, R0                      ; 8033
                28   AE           50                30  81 00239        ADDB3   #48, R0, HEXSTRING                      ;
                                                    6E  DD 0023E        PUSHL   REND                                    ; 8034
                                                47  A8  9F 00240        PUSHAB  71(TPTR)                                ;
                                                30  AE  9F 00243        PUSHAB  HEXSTRING                               ;
                                                    01  DD 00246        PUSHL   #1                                      ;
                                                    59  DD 00248        PUSHL   DLEPTR                                  ;
                          0000V  CF                 05  FB 0024A        CALLS   #5, REGISTER_FILL                       ;
          50         52           01                01  EF 0024F        EXTZV   #1, #1, PSLVAL, R0                      ; 8035
```

```
                28  AE              50          30  81 00254        ADDB3   #48, R0, HEXSTRING
                                                6E  DD 00259        PUSHL   REND                        8036
                                        4B      A8  9F 0025B        PUSHAB  75(TPTR)
                                        30      AE  9F 0025E        PUSHAB  HEXSTRING
                                                01  DD 00261        PUSHL   #1
                                                59  DD 00263        PUSHL   DLEPTR
                            0000V       CF          05  FB 00265   CALLS   #5, REGISTER_FILL
          50                        52  01          00  EF 0026A   EXTZV   #0, #1, PSLVAL, R0          8037
                28  AE              50          30  81 0026F        ADDB3   #48, R0, HEXSTRING
                                                6E  DD 00274        PUSHL   REND                        8038
                                        4F      A8  9F 00276        PUSHAB  79(TPTR)
                                        30      AE  9F 00279        PUSHAB  HEXSTRING
                                                01  DD 0027C        PUSHL   #1
                                                59  DD 0027E        PUSHL   DLEPTR
                            0000V       CF          05  FB 00280   CALLS   #5, REGISTER_FILL
                                                    04  0C285       RET                             8045

; Routine Size:  646 bytes,    Routine Base:  DBG$CODE + 40DE
```

```
;  7973     8046  1 ROUTINE REGISTER_FILL(DLEPTR, LENGTH, SRCPTR, DSTPTR, REND): NOVALUE =
;  7974     8047  1 !
;  7975     8048  1 ! FUNCTION
;  7976     8049  1 !       This routine fills a source string, usually with a hexadecimal register
;  7977     8050  1 !       value, into a field in a register screen display.  In addition to
;  7978     8051  1 !       simply filling in the source characters into the destination location,
;  7979     8052  1 !       it checks to see if the field is changing value.  If so, the contents
;  7980     8053  1 !       of the field is marked to be put out in the rendition specified by the
;  7981     8054  1 !       REND input parameter (usually bolding).  Hence the register display
;  7982     8055  1 !       highlights changed register values by showing their new values in a
;  7983     8056  1 !       different rendition.
;  7984     8057  1 !
;  7985     8058  1 ! INPUTS
;  7986     8059  1 !       DLEPTR  - A pointer to the Display Line Entry for the text line into
;  7987     8060  1 !                 which the new source string is to be filled.
;  7988     8061  1 !
;  7989     8062  1 !       LENGTH  - The length of the character string to be filled in.
;  7990     8063  1 !
;  7991     8064  1 !       SRCPTR  - A pointer to the source location containing the character
;  7992     8065  1 !                 string to be copied.
;  7993     8066  1 !
;  7994     8067  1 !       DSTPTR  - A pointer to the destination location into which the
;  7995     8068  1 !                 character string is to be copied.
;  7996     8069  1 !
;  7997     8070  1 !       REND    - The rendition codes to be used for fields which have
;  7998     8071  1 !                 changed their values.
;  7999     8072  1 !
;  8000     8073  1 ! OUTPUTS
;  8001     8074  1 !       NONE
;  8002     8075  1 !
;  8003     8076  1
;  8004     8077  2     BEGIN
;  8005     8078  2
;  8006     8079  2     MAP
;  8007     8080  2         DLEPTR: REF DBG$DLINE_ENTRY;    ! Pointer to the Display Line Entry
;  8008     8081  2
;  8009     8082  2     LOCAL
;  8010     8083  2         TPTR: REF VECTOR[,BYTE];        ! Pointer to display line's text vector
;  8011     8084  2
;  8012     8085  2
;  8013     8086  2
;  8014     8087  2     ! If the new character string is identical in value to the string that is
;  8015     8088  2     ! already in the destination field, we return immediately.  This leaves
;  8016     8089  2     ! the string as is and it leaves the rendition as normal video.
;  8017     8090  2     !
;  8018     8091  2     IF CH$EQL(.LENGTH, .SRCPTR, .LENGTH, .DSTPTR, 0) THEN RETURN;
;  8019     8092  2
;  8020     8093  2
;  8021     8094  2     ! The character string is changing value.  Hence we copy the new value to
;  8022     8095  2     ! the destination address and we mark its rendition as specified by the
;  8023     8096  2     ! REND parameter.  Then return.
;  8024     8097  2     !
;  8025     8098  2     CH$MOVE(.LENGTH, .SRCPTR, .DSTPTR);
;  8026     8099  2     DLEPTR[DBG$V_DLINE_RENDFLG] = TRUE;
;  8027     8100  2     TPTR = DLEPTR[DBG$A_DLINE_TEXT];
;  8028     8101  2     CH$FILL(.REND, .LENGTH, TPTR[1 + .TPTR[0] + (.DSTPTR - TPTR[1])]);
;  8029     8102  2     RETURN;
```

```
; 8030          8103  2
; 8031          8104  1     END;


                                                003C 00000 REGISTER_FILL:
                                                           .WORD   Save R2,R3,R4,R5                                    ; 8046
                    10   BC      0C   BC      08   AC  29 00002          CMPC3   LENGTH, @SRCPTR, @DSTPTR                ; 8091
                                                AC  26 13 00009          BEQL    1$                                      ;
                    10   BC      0C   BC      08   AC  28 0000B          MOVC3   LENGTH, @SRCPTR, @DSTPTR                ; 8098
                                         50      04   AC  D0 00012          MOVL    DLEPTR, R0                          ; 8099
                                    09   A0          01  88 00016          BISB2   #1, 9(R0)                           ;
                                         50          0C  C0 0001A          ADDL2   #12, TPTR                           ; 8100
                                         51          60  9A 0001D          MOVZBL  (TPTR), R1                          ; 8101
                        52       10   AC          50  C3 00020          SUBL3   TPTR, DSTPTR, R2                    ;
                                         51          52  C0 00025          ADDL2   R2, R1                              ;
          08   AC        14   AC              6E      00  2C 00028          MOVC5   #0, (SP), REND, LENGTH, (R1)[TPTR]  ;
                                                  6140      0002F                                                       ;
                                                          04 00031 1$:      RET                                         ; 8104

; Routine Size:  50 bytes,    Routine Base:  DBG$CODE + 4364
```

```
;  8033     8105  1 ROUTINE WINDOW_SOURCE_LINE(DISPID, DOWN_FLAG) =
;  8034     8106  1 !
;  8035     8107  1 ! FUNCTION
;  8036     8108  1 !       This routine returns the Display Line Entry which contains the first
;  8037     8109  1 !       source line number of the current screen window of a specified display.
;  8038     8110  1 !       Normally, this is simply the line number of the first Display Line
;  8039     8111  1 !       Entry in the display's screen window.  However, if that first line does
;  8040     8112  1 !       not have a line number (it could be an unnumbered page eject record in
;  8041     8113  1 !       the source file), then this routine searches forward and backward until
;  8042     8114  1 !       it finds the nearest source line with a line number.  A pointer to the
;  8043     8115  1 !       Display Line Entry for that source line is returned as this routine's
;  8044     8116  1 !       value.  This routine is called when scrolling source displays.
;  8045     8117  1 !
;  8046     8118  1 ! INPUTS
;  8047     8119  1 !       DISPID  - A pointer to the Screen Display Entry for which the first
;  8048     8120  1 !                 window line number is desired.
;  8049     8121  1 !
;  8050     8122  1 !       DOWN_FLAG - A flag set to TRUE if the display is to be scrolled down
;  8051     8123  1 !                 and set to FALSE if the display is to be scrolled up.
;  8052     8124  1 !
;  8053     8125  1 ! OUTPUTS
;  8054     8126  1 !       This routine returns a pointer to the Screen Display Line Entry for
;  8055     8127  1 !                 the source line with the first source line number in the
;  8056     8128  1 !                 current display window.
;  8057     8129  1 !
;  8058     8130  1
;  8059     8131  2     BEGIN
;  8060     8132  2
;  8061     8133  2     MAP
;  8062     8134  2         DISPID: REF DBG$DISP_ENTRY;     ! Pointer to Screen Display Entry
;  8063     8135  2
;  8064     8136  2     LOCAL
;  8065     8137  2         DLEPTR: REF DBG$DLINE_ENTRY;    ! Pointer to current Display Line Entry
;  8066     8138  2
;  8067     8139  2
;  8068     8140  2
;  8069     8141  2     ! Search the specified display's Display Line Entry list in the preferred
;  8070     8142  2     ! direction until we find a Display Line Entry with a line number.  When
;  8071     8143  2     ! we find such an entry, return its address to the caller.  The preferred
;  8072     8144  2     ! direction is forward if we are scrolling down and backward if we are
;  8073     8145  2     ! scrolling up.
;  8074     8146  2     !
;  8075     8147  2     DLEPTR = .DISPID[DBG$L_DISP_WINDOW_PTR];
;  8076     8148  2     WHILE TRUE DO
;  8077     8149  3         BEGIN
;  8078     8150  3         IF .DLEPTR[DBG$L_DLINE_LINUM] GEQ 0 THEN RETURN .DLEPTR;
;  8079     8151  3         IF .DOWN_FLAG
;  8080     8152  3         THEN
;  8081     8153  4             BEGIN
;  8082     8154  4             IF .DLEPTR EQL .DISPID[DBG$L_DISP_END_LINE_PTR] THEN EXITLOOP;
;  8083     8155  4             DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  8084     8156  4             END
;  8085     8157  4
;  8086     8158  3         ELSE
;  8087     8159  4             BEGIN
;  8088     8160  4             IF .DLEPTR EQL .DISPID[DBG$L_DISP_START_LINE_PTR] THEN EXITLOOP;
;  8089     8161  4             DLEPTR = .DLEPTR[DBG$L_DLINE_BLINK];
```

```
;  8090          8162  3              END;
;  8091          8163  3
;  8092          8164  2          END;
;  8093          8165  2
;  8094          8166  2
;  8095          8167  2      ! We did not find a Display Line Entry with a line number scanning in the
;  8096          8168  2      ! preferred direction.  Hence we now scan the Display Line Entry list in
;  8097          8169  2      ! the opposite direction (backward if scrolling down and forward if scroll-
;  8098          8170  2      ! ing up).  When we find an entry with a line number, we return its address
;  8099          8171  2      ! to the caller.
;  8100          8172  2      !
;  8101          8173  2      DLEPTR = .DISPID[DBG$L_DISP_WINDOW_PTR];
;  8102          8174  2      WHILE TRUE DO
;  8103          8175  3          BEGIN
;  8104          8176  3          IF .DLEPTR[DBG$L_DLINE_LINUM] GEQ 0 THEN RETURN .DLEPTR;
;  8105          8177  3          IF .DOWN_FLAG
;  8106          8178  3          THEN
;  8107          8179  4              BEGIN
;  8108          8180  4              IF .DLEPTR EQL .DISPID[DBG$L_DISP_START_LINE_PTR] THEN EXITLOOP;
;  8109          8181  4              DLEPTR = .DLEPTR[DBG$L_DLINE_BLINK];
;  8110          8182  4              END                                                                 2
;  8111          8183  4
;  8112          8184  3          ELSE
;  8113          8185  4              BEGIN
;  8114          8186  4              IF .DLEPTR EQL .DISPID[DBG$L_DISP_END_LINE_PTR] THEN EXITLOOP;
;  8115          8187  4              DLEPTR = .DLEPTR[DBG$L_DLINE_FLINK];
;  8116          8188  3              END;
;  8117          8189  3
;  8118          8190  2          END;
;  8119          8191  2
;  8120          8192  2
;  8121          8193  2      ! If we have not found a Display Line Entry with a line number, something
;  8122          8194  2      ! is wrong and we signal an internal DEBUG error.
;  8123          8195  2      !
;  8124          8196  2      $DBG_ERROR('DBGSCREEN\WINDOW_SOURCE_LINE');
;  8125          8197  2      RETURN 0;
;  8126          8198  2
;  8127          8199  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

44 4E 49 57 5C 4E 45 45 52 43 53 47 42 44 1C  00A19 P.AIN:    .ASCII  <28>\DBGSCREEN\<92>\WINDOW_SOURCE_LINE\     ;
   45 4E 49 4C 5F 45 43 52 55 4F 53 5F 57 4F  00A28                                                          ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                          0000 00000 WINDOW_SOURCE_LINE:
                                                     .WORD   Save nothing                                   ; 8105
                                51        04   AC  D0 00002            MOVL    DISPID, R1                  ; 8147
                                50        28   A1  D0 00006            MOVL    40(R1), DLEPTR              ;
                                          10   A0  D5 0000A 1$:         TSTL    16(DLEPTR)                  ; 8150
                                          56   18 00000D               BGEQ    7$                          ;
                                08        08   AC  E9 0000F            BLBC    DOWN_FLAG, 2$               ; 8151
```

```
                    24   A1              50  D1 00013          CMPL    DLEPTR, 36(R1)                  ; 8154
                                         11  13 00017          BEQL    3$
                         50              60  D0 00019          MOVL    (DLEPTR), DLEPTR                ; 8155
                                         EC  11 0001C          BRB     1$                              ; 8151
                    20   A1              50  D1 0001E 2$:      CMPL    DLEPTR, 32(R1)                  ; 8160
                                         06  13 00022          BEQL    3$
                         50         04   A0  D0 00024          MOVL    4(DLEPTR), DLEPTR               ; 8161
                                         E0  11 00028          BRB     1$                              ; 8148
                         50         28   A1  D0 0002A 3$:      MOVL    40(R1), DLEPTR                  ; 8173
                                    10   A0  D5 0002E 4$:      TSTL    16(DLEPTR)                      ; 8176
                                         32  18 00031          BGEQ    7$
                    0C   08              AC  E9 00033          BLBC    DOWN_FLAG, 5$                   ; 8177
                    20   A1              50  D1 00037          CMPL    DLEPTR, 32(R1)                  ; 8180
                                         11  13 0003B          BEQL    6$
                         50         04   A0  D0 0003D          MOVL    4(DLEPTR), DLEPTR               ; 8181
                                         EB  11 00041          BRB     4$                              ; 8177
                    24   A1              50  D1 00043 5$:      CMPL    DLEPTR, 36(R1)                  ; 8186
                                         05  13 00047          BEQL    6$
                         50              60  D0 00049          MOVL    (DLEPTR), DLEPTR                ; 8187
                                         E0  11 0004C          BRB     4$                              ; 8174
                              00000000'  EF  9F 0004E 6$:      PUSHAB  P.AIN                           ; 8196
                                         01  DD 00054          PUSHL   #1
                              00028362   8F  DD 00056          PUSHL   #164706
          00000000G   00                 03  FB 0005C          CALLS   #3, LIB$SIGNAL
                                         50  D4 00063          CLRL    R0                              ; 8197
                                             04 00065 7$:      RET                                     ; 8199

; Routine Size:  102 bytes,    Routine Base:  DBG$CODE + 4396

; 8128          8200  1
; 8129          8201  0 END ELUDOM


                                                               .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$PLIT                      2614  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$GLOBAL                      32  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$OWN                       6004  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$CODE                     17404  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                           Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619        18         0        1000        00:01.8
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1     32         3         9           7        00:00.1
```

```
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1          1545       146         9        97      00:01.8
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                                418         2         0        31      00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1           386        14         3        22      00:00.3



;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:DBGSCREEN/OBJ=OBJ$:DBGSCREEN MSRC$:DBGSCREEN/UPDATE=(ENH$:DBGSCREEN)

; Size:         17404 code + 8650 data bytes
; Run Time:        04:38.6
; Elapsed Time:    05:29.1
; Lines/CPU Min:    1765
; Lexemes/CPU-Min: 16483
; Memory Used:  763 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGSCREEN
LIS

DBGPRINT
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGSSV
LIS

DBGSTEPGO
LIS

DBGSTART
LIS

DBGSOURCE
LIS